Towards Building a Global Oracle: A Physical Mashup using Artificial Intelligence Technology

Carolina Fortuna, Blaz Fortuna, Matevz Vucnik, Klemen Kenda, Alexandra Moraru and Dunja Mladenic

http://sensorlab.ijs.si/

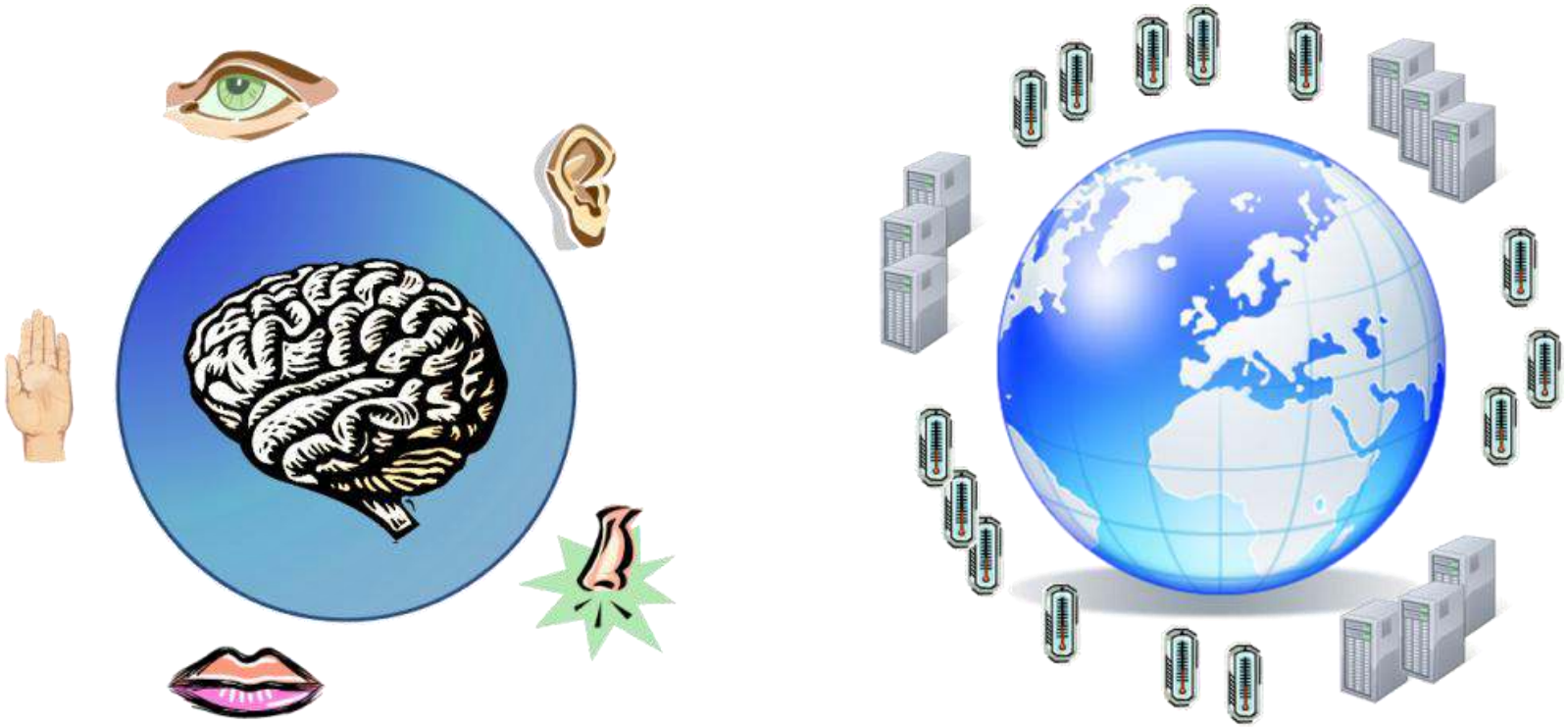
http://sensors.ijs.si/

Web of Things Workshop, June 19, 2012.

Outline

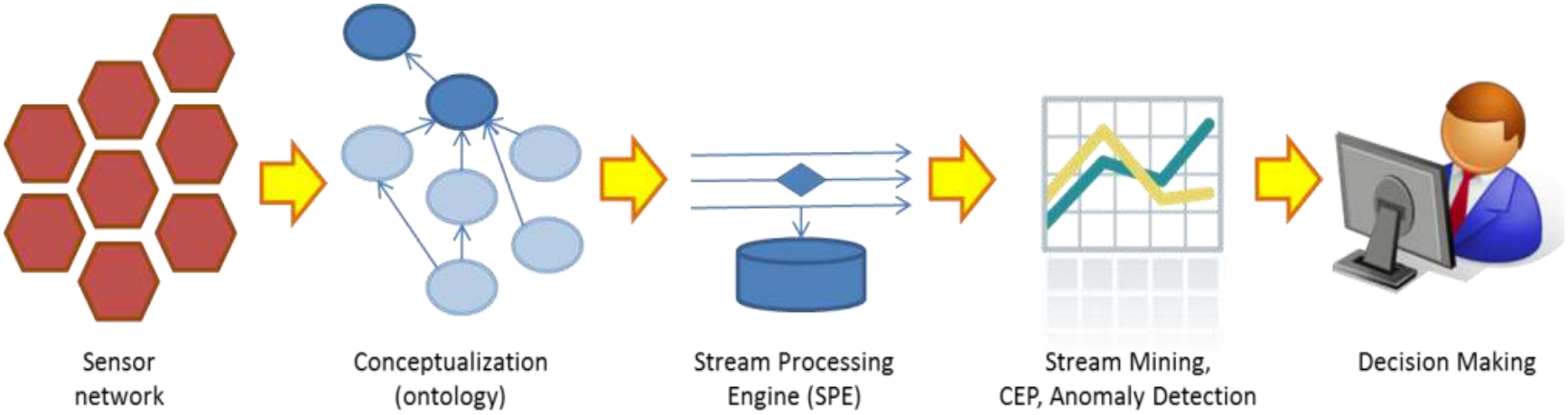
- The big picture
- System architecture
- System Implementation
  - Sensor Data Acquisition
  - SenseStream
  - The Videk mash-up server
  - Videk API and GUI
- External data sources
  - ResearchCyc
  - Web Services
- Conclusions

The big picture



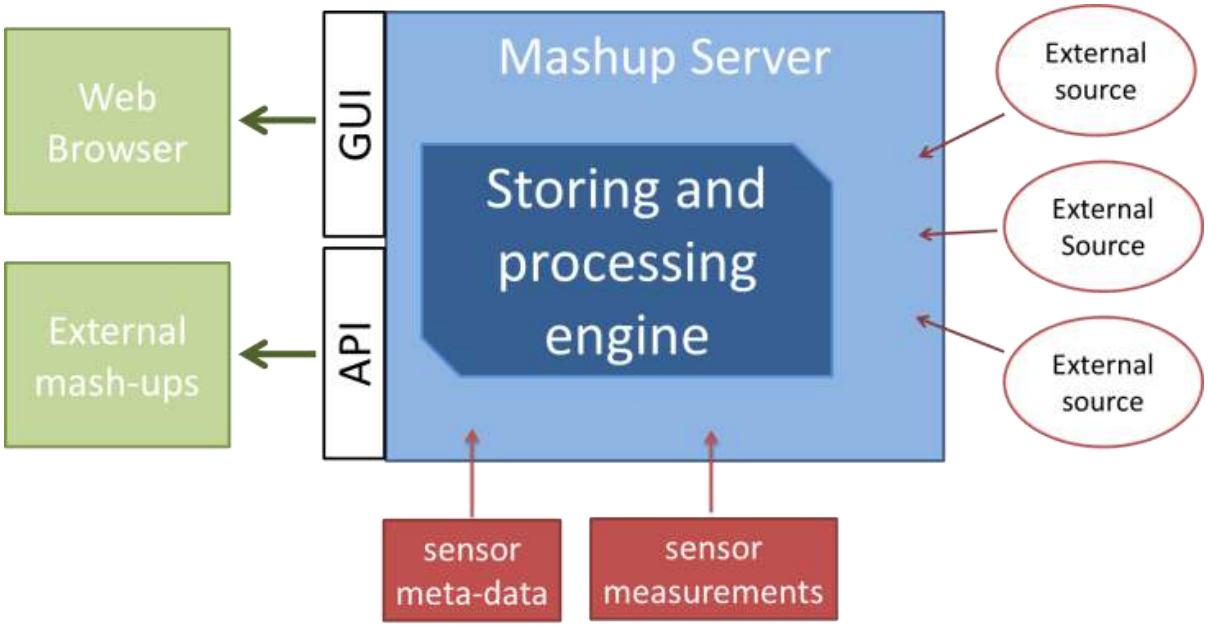
Analogy between humans and WoT

The big picture – AI technologies for physical mash-ups



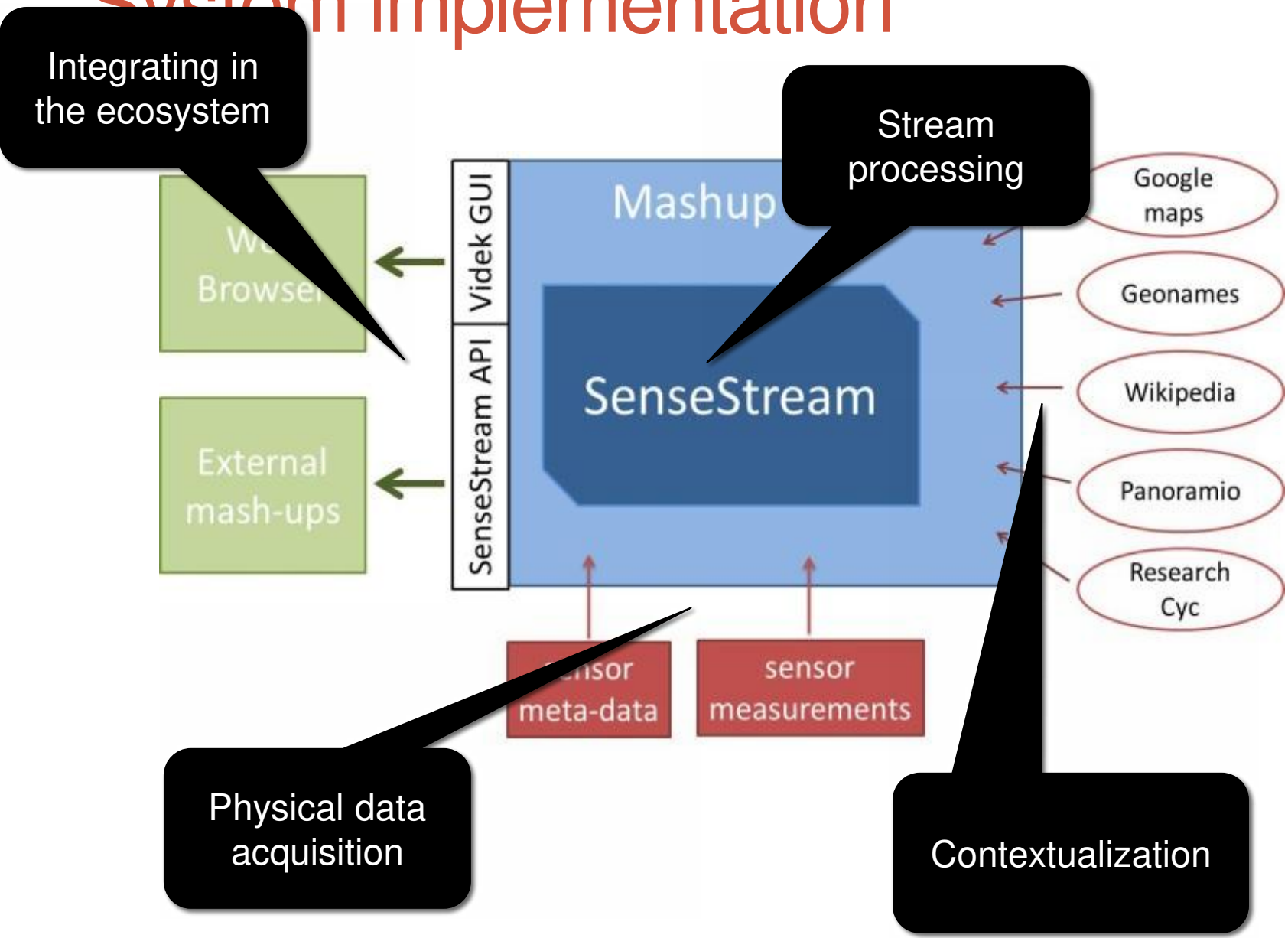
- WoT mashups can take and use the data at any of these stages.

System architecture

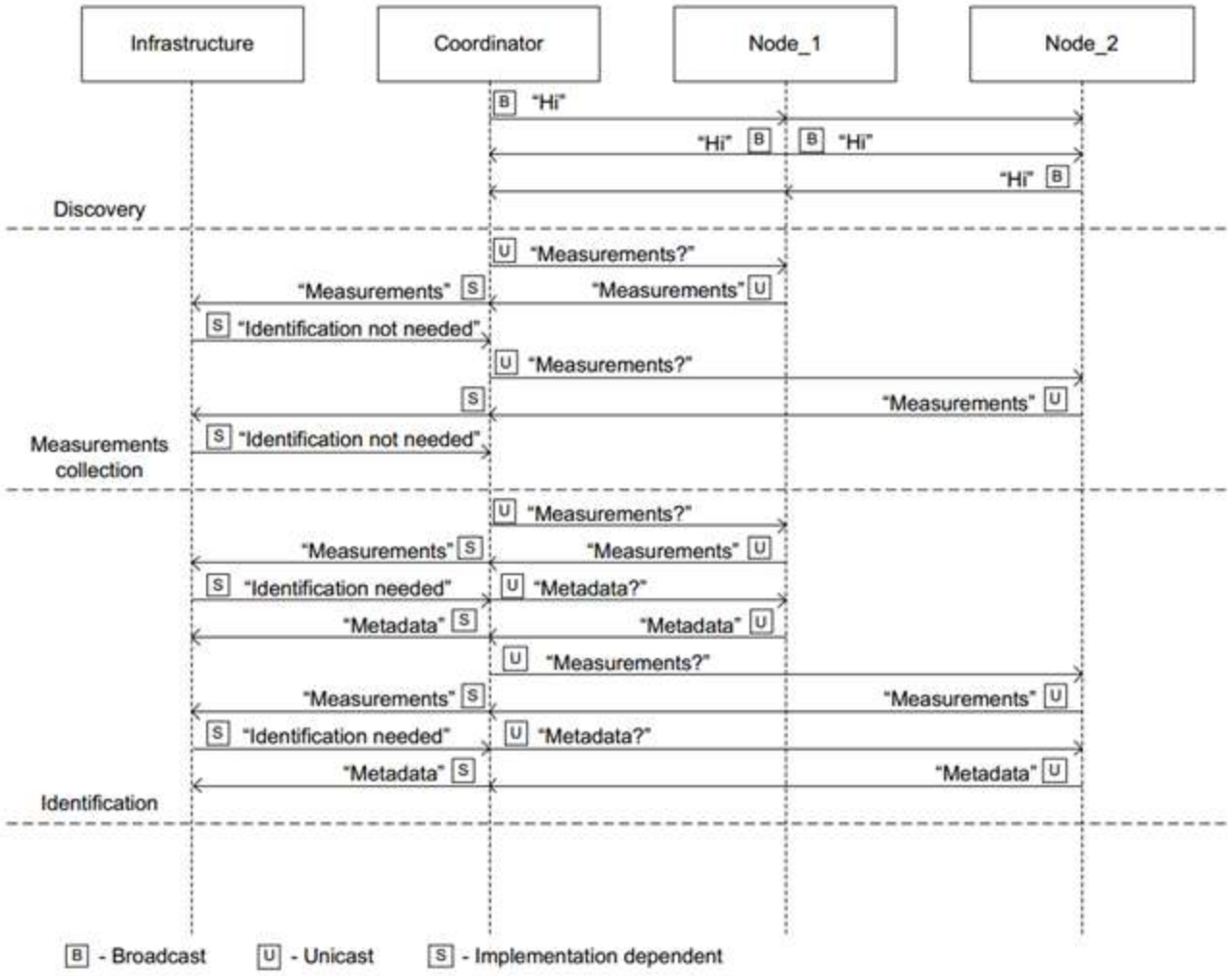


- Mashup server containing a stream storing and processing engine
- Sensor data
- External web services
- GUI/API

System implementation

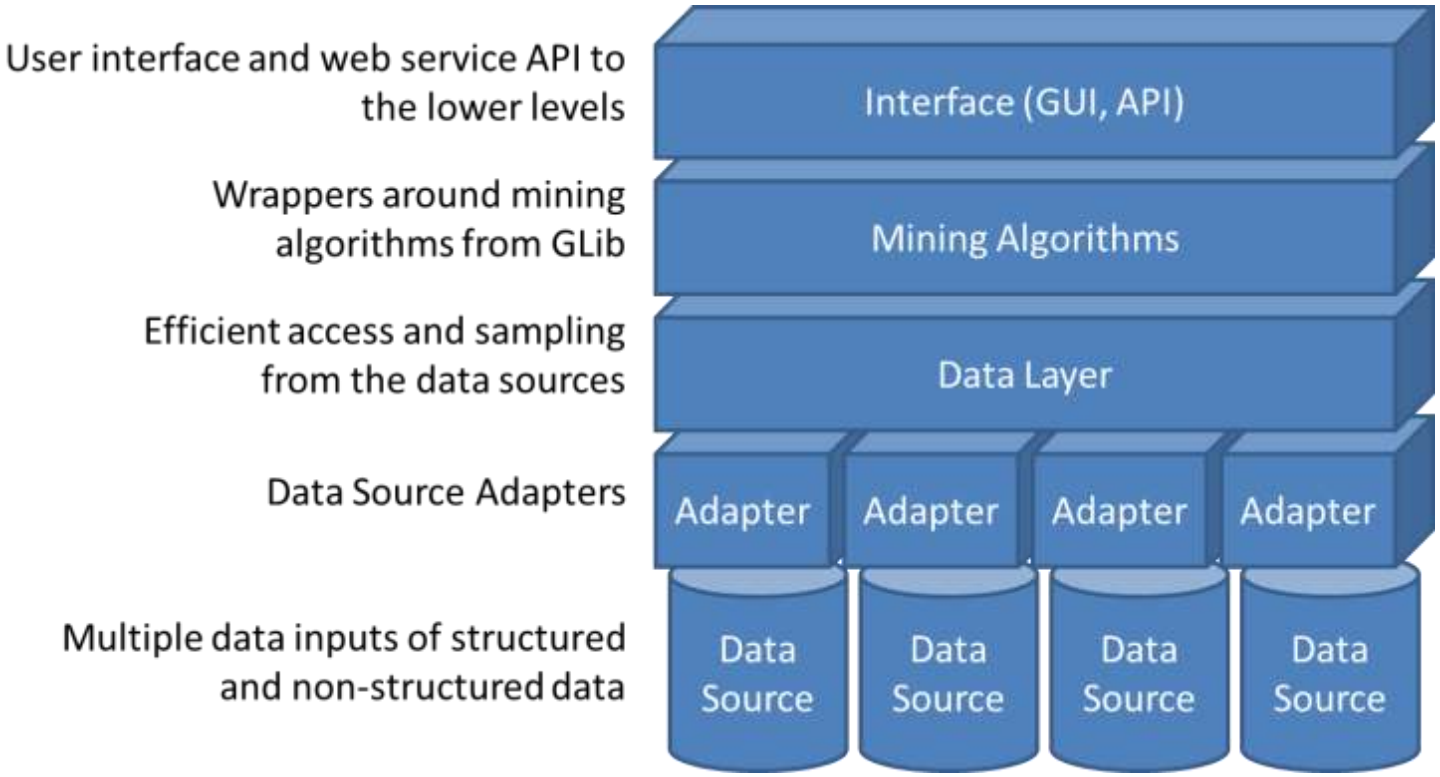


Sensor data acquisition



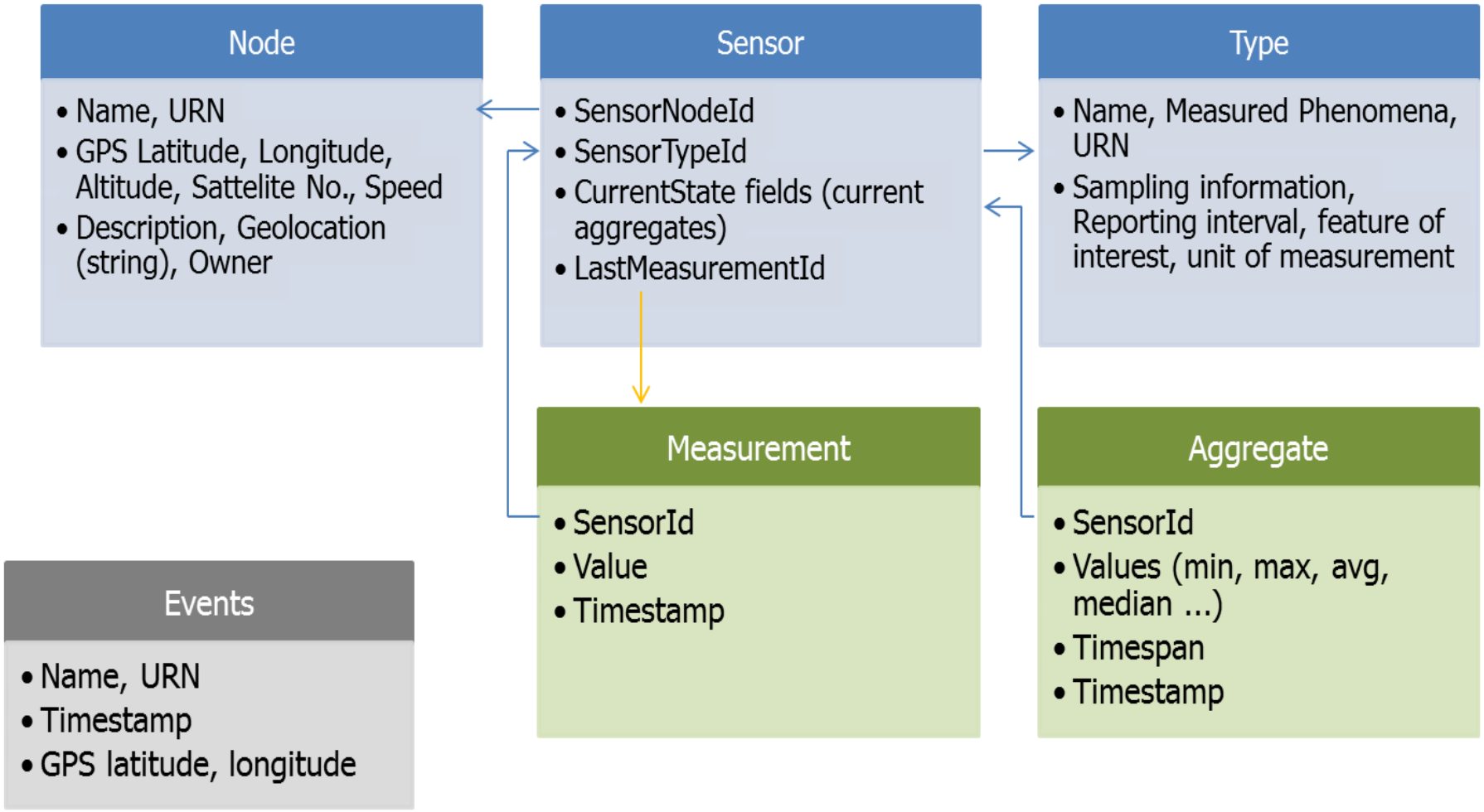
Device Identification Protocol (DIP) sequence diagram

SenseStream



- stream mining and event detection engine
- The central part of the system
  - Data Layer and
  - Mining Algorithms

SenseStream data layer schema



Current State & Aggregates

- CurrentState is partly stand-alone object (configuration),
- partly included (distributed within) in Sensor store records.
- Stores current aggregates – state of the system
- Saves aggregates to the aggregate store after transition into a new time window

Calculate on-line (update with last measurement)

- Count
- Average
- Sum
- Min
- Max
- Standard deviation*

Calculate directly on the series

- Median
- 1st Quartile
- 3rd Quartile

Aggregates

Primary aggregates

Calculated from raw measurements, fine grained.

Secondary aggregates

Calculated from other aggregates (only possible to use with on-line type).

Configure time spans, aggregate types, aggregates for certain sensor types, aggregates for certain sensor.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <timespans>
    <timespan id="1" timewindow="3600" />
    <timespan id="2" pid="1" timewindow="24" interval="1"/>
    <timespan id="3" pid="2" timewindow="7" interval="1"/>
    <timespan id="3" pid="2" timewindow="30" interval="1"/>
    <timespan id="4" pid="2" timewindow="365" interval="1"/>
  </timespans>
  <aggregates>
    <aggregate type="MAX"/>
    <aggregate type="MIN">
      <timespan id="1" timewindow="3600">
      <timespan id="2" pid="1" timewindow="48" interval="24"/>
    </aggregate>
    <aggregate type="AVG"/>
    <aggregate type="SUM"/>
    <aggregate type="STD"/>
    <aggregate type="MED"/>
    <aggregate type="1QU"/>
    <aggregate type="3QU"/>
    <aggregate type="CNT"/>
  </aggregates>
  <sensortypes>
    <sensortype id="1">
      <aggregate type="MAX"/>
      <aggregate type="SUM"/>
    </sensortype>
  </sensortypes>
  <sensors>
    <sensor id="1">
      <aggregate type="MAX"/>
      <aggregate type="SUM"/>
    </sensor>
  </sensors>
</configuration>
```

Time windows & intervals

- Time windows of aggregates can overlap
- Overlapping interval is set in configuration file (interval)
- For example:
  - Weekly aggregates can be calculated from Monday to Monday, from Tuesday to Tuesday, etc.

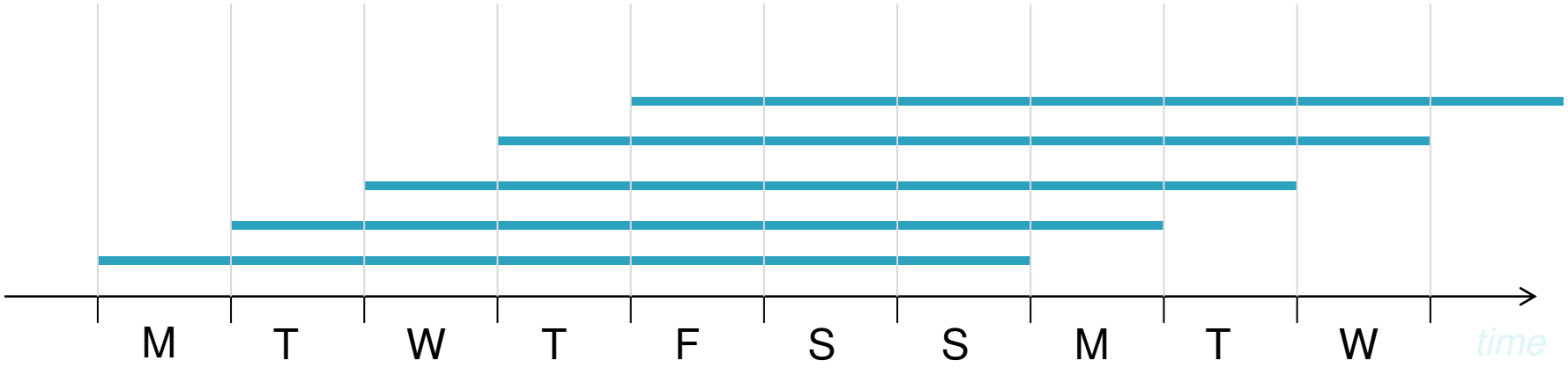
```
<timespan id="1" timewindow="3600" />

<timespan id="2" pid="1"
 timewindow="24" interval="1"/>

<timespan id="3" pid="2"
 timewindow="7" interval="1"/>
```

7-day time window

Overlap/update interval is 1 day



Event Detection

- Easy and fast detection of events on current state data (very simple rules)
- Simple validation of more complex event queries (using current state and previous aggregates)
- Time can be handled with ease ...

Fog forming example

If

(humidity[AVG,1h] < 90%) &
(humidity[AVG,10m] > 95%)

Then

trigger fog forming risk event.

Road Icing example

If

(precipitation[SUM,12h,6h ago] > X) &
(temparature[MAX,12h,6h ago] > 0) &
(temperature[MIN, 6h]) < 0)

Then

trigger road icing risk event.

Time example

If

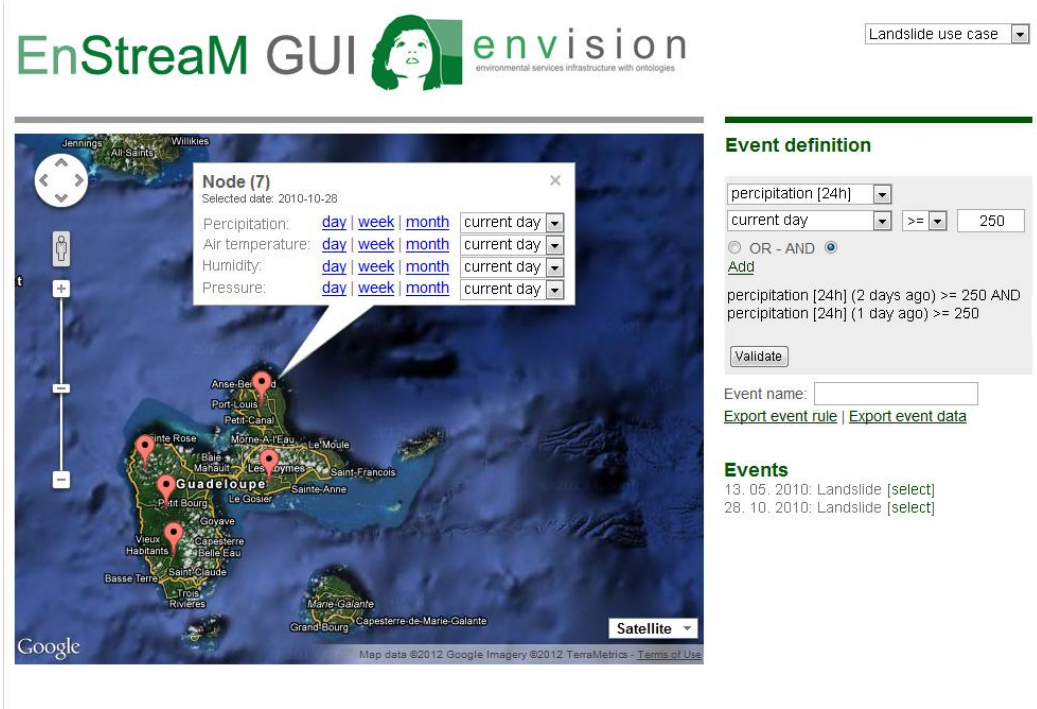
(temperature[AVG,1w,3d ago] <-5) &
(temperature[AVG,24h,2d ago] < 5) &
(temperature[AVG,24h,1d ago] < 5) &

Then

trigger lake still frozen event.

Rule generation and export

```
<And> <Atom>
  <op> <Rel iri="cyc:sensorObservation"/> </op>
  <Var> sensor </Var>
  <Ind iri="cyc:Raindrop"/> </Atom>
<Atom>
  <op> <Rel iri="cyc:doneBy"/> </op>
  <Var> sensor </Var>
  <Var> measurement </Var> </Atom>
<Atom>
  <op> <Rel iri="cyc:measurementResult"/>
</op>
  <Var> measurement </Var>
  <Var> val1 </Var> </Atom>
<Atom>
  <op> <Rel iri="cyc:duration"/> </op>
  <Var> measurement </Var>
                <Ind
type="xs:time">24:00:00</Ind> </Atom>
<Atom>
  <op> <Rel iri="cyc:greaterThan"/> </op>
  <Var> val1 </Var>
  <Ind type="xs:float">250</Ind>
</Atom> </And>
```



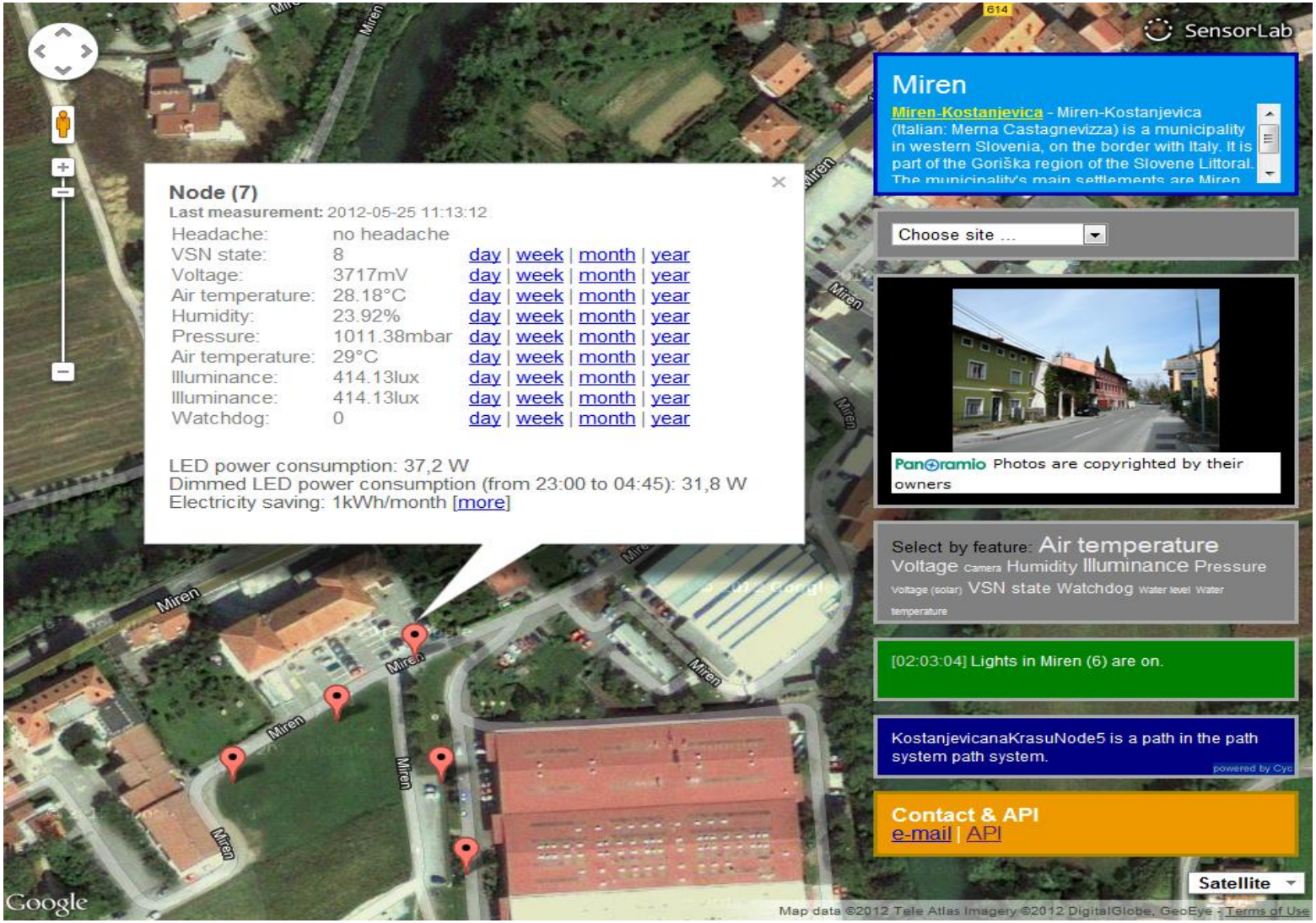
The Videk mash-up server

- acts as a glue between different components and services used
- it interfaces with sensors receiving data from these
- parses and multiplexes the data
  - to the back-up database,
  - to SenseStream,
  - to ResearchCyc,
  - to the triple store.
- it exposes the API to be used by external applications
- provides a GUI with widgets which mashes up the data and resulting knowledge

Videk API

- http://sensors.ijs.si/xml/current-state the XML structured answer includes:
- **nodes**
  - records (int) - *number of nodes*
  - **node**
    - id (int) - *internal id of the node*
    - name (string) - *identifier of the node*
    - latitude (double) - *latitude of the node*
    - longitude (double) - *longitude of the node*
    - **sensor**
      - id (int) - *internal id of the sensor*
      - sensortypeid (int) - *internal id of the type of the sensor*
      - featureofmeasurement (string) - *human readable feature of measurement*
      - unitofmeasurement (string) - *human readable unit of measurement*
      - lastmeasurement (double) - *value of the last measurement from the sensor*
      - measurementtime (time) - *MySQL fromatted timestamp of the measurement*

Videk GUI



http://sensors.ijs.si/

External data sources

- Research Cyc for structured knowledge representation and reasoning
- Open web services for additional context collection
  - Based on the sensor nodes' GPS coordinates, Google maps are used as the GUI's background and the right focus on the deployment locations is presented.
  - Based on the GPS coordinates, the Geonames service is used to retrieve the name of the place where the sensors are deployed and relevant information from Wikipedia.
  - Finally, the Panoramio service is invoked to retrieve and render pictures of the surrounding area.

ResearchCyc

- the research release of Cyc, an artificial intelligence system
  - comprised of a knowledge base and a reasoning engine
- the idea behind it is to encode knowledge in a structured way and reason about it similar to the way the human mind does it.
  - each of us learns the concept of a tree, branch, leaf and fruit.
  - we learn relationships between these concepts: that a tree has branches, on a branch grow leaves and fruit.
  - we are able to recognize instances of these: this apple tree, this apple, this apple leaf.
- This knowledge builds up in our brains over years and makes it possible to understand, communicate and reason.

ResearchCyc

Individual : VicNode1 (+) CURE

on the term

isa : ElectronicDevice
isa : Computer
connectedTo : virtualsensor3VicNode1 TSL2561VicNode1 sht11VicNode1
(connectedTo scp1000VicNode1 VicNode1)
(connectedTo virtualsensor2VicNode1 VicNode1)
(connectedTo virtualsensor1VicNode1 VicNode1)
(deviceUsed Testing VicNode1)
hasDevices : virtualsensor3VicNode1 TSL2561VicNode1 scp1000VicNode1 sht11VicNode1 virtualsensor2VicNode1 virtualsensor1VicNode1
latitude : (Degree-UnitOfAngularMeasure 46.042873)
longitude : (Degree-UnitOfAngularMeasure 14.487469)
nameString : "VicNode1"
objectFoundInLocation : IndoorMounting Vic
physicalParts : virtualsensor3VicNode1 TSL2561VicNode1 scp1000VicNode1 sht11VicNode1 virtualsensor2VicNode1 virtualsensor1VicNode1
(queryHasVeryHighPertinenceForThing GetLinkToMap VicNode1)
supportedBy : VicBuilding1

Mt : BaseKB
(isa VicNode1 ElectronicDevice)

English Translation :
Vic Node 1 is an electronic device.

- We inserted the collected meta-data in the KB
- Using NL generation rules we provide transliteration of the logical structures.

Summary

- made an analogy between human senses and sensors; and between human brain and artificial intelligence technology
  - this analogy leads to the concept of Global Oracle.
- described Videk - a physical mashup which uses artificial intelligence technology
  - Videk automatically collects data from sensors.
  - the data is processed and stored by SenseStream while the meta-data is fed into ResearchCyc.
  - SenseStream indexes aggregates, performs clustering and learns rules which then it exports as RuleML.
  - ResearchCyc performs logical inference on the meta-data and transliterates logical sentences.
  - The GUI mashes up sensor data with SenseStream output, ResearchCyc output and other external data sources: GoogleMaps, Geonames, Wikipedia and Panoramio.

Future work

- extend SenseStream and couple it with ResearchCyc.
- StreamSense
  - add additional mining and learning algorithms which make sense of the input sensor data.
  - implement detection of events that comply with the learned rules that are currently exported as RuleML.
  - the learned rules, together with richer meta-data automatically collected from upcoming sensor deployments will be also be inserted into ResearchCyc for reasoning and transliteration.
- all developments will reflect on the GUI and API in time.

Lab: http://sensorlab.ijs.si/

WoT Mash-up: http://sensors.ijs.si/

Tutorial: http://carolinafortuna.com/web-of-things-tutorial/

Contact: carolina.fortuna@ijs.si