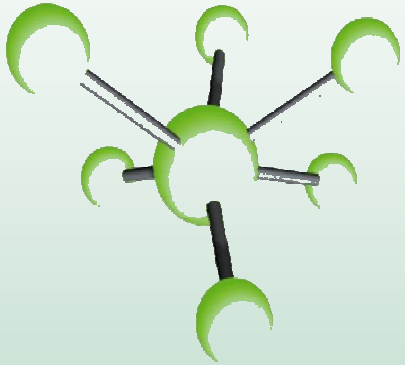
AUSTRIAN INSTITUTE OF TECHNOLOGY
TOMORROW TODAY

S@NY
Sensors Anywhere
FP6-033564



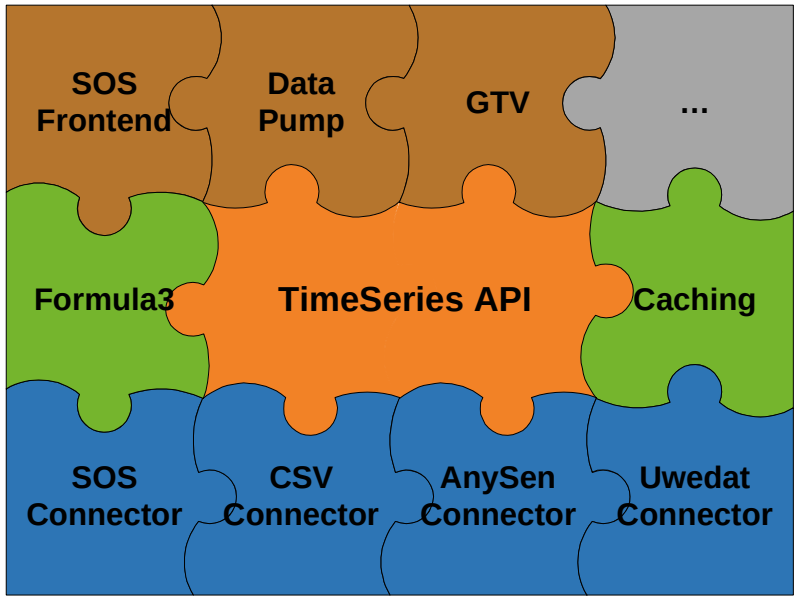
FOSS4G 2009 - Sydney

The Time Series Toolbox

Thomas Bleier, Arndt Bonitz, Bojan Božić,
Thomas Ponweiser

The Time Series Toolbox

- A set of API‘s and software components
- For building services and applications
- That work with time series data (record, process, store, publish, etc.)



Design Goals

- Modular architecture
  use components as you like, not „all-or-nothing“
- Standards-based
  e.g. OGC Sensor Observation Service
- Flexible design
  From single float values to complex data structures
- Extensible
  Well-defined interfaces allow easy extension

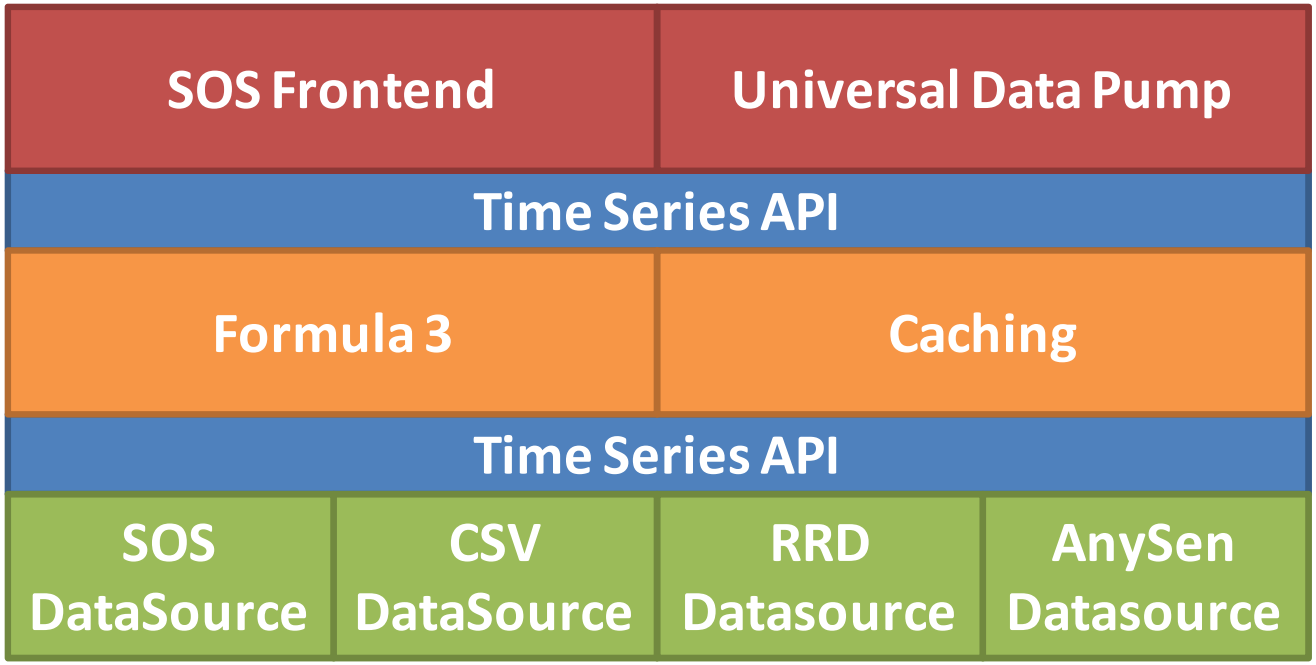
Environment / Requirements

- Main language: Java 1.6
- One component written in Python (→ Jython)
- Can be used in all types of applications:
  - Webservices (J2EE, etc)
  - Command-line applications
  - Desktop GUI applications

S@NY Sensors Anywhere

Layered Architecture

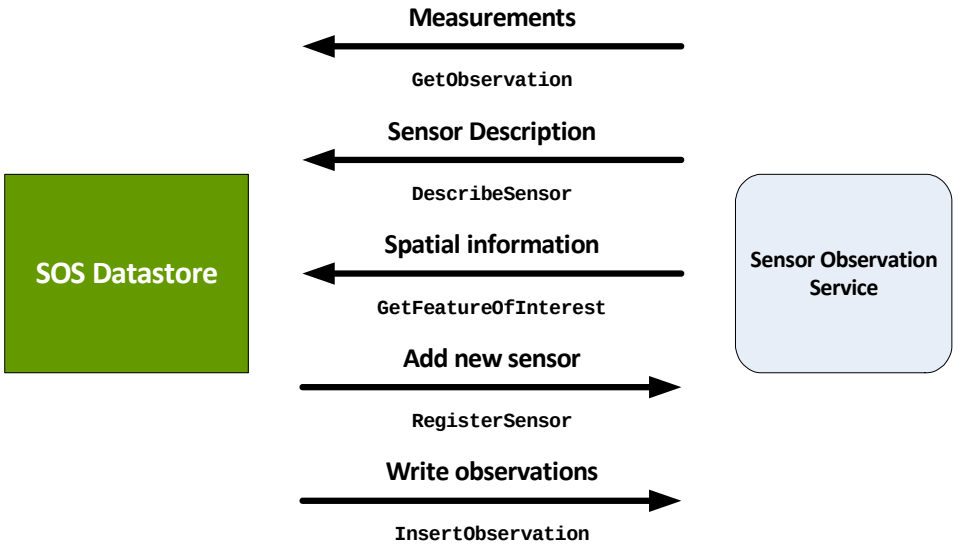
- Data connector components
- Core processing components
- Frontend components



Information Society Technologies

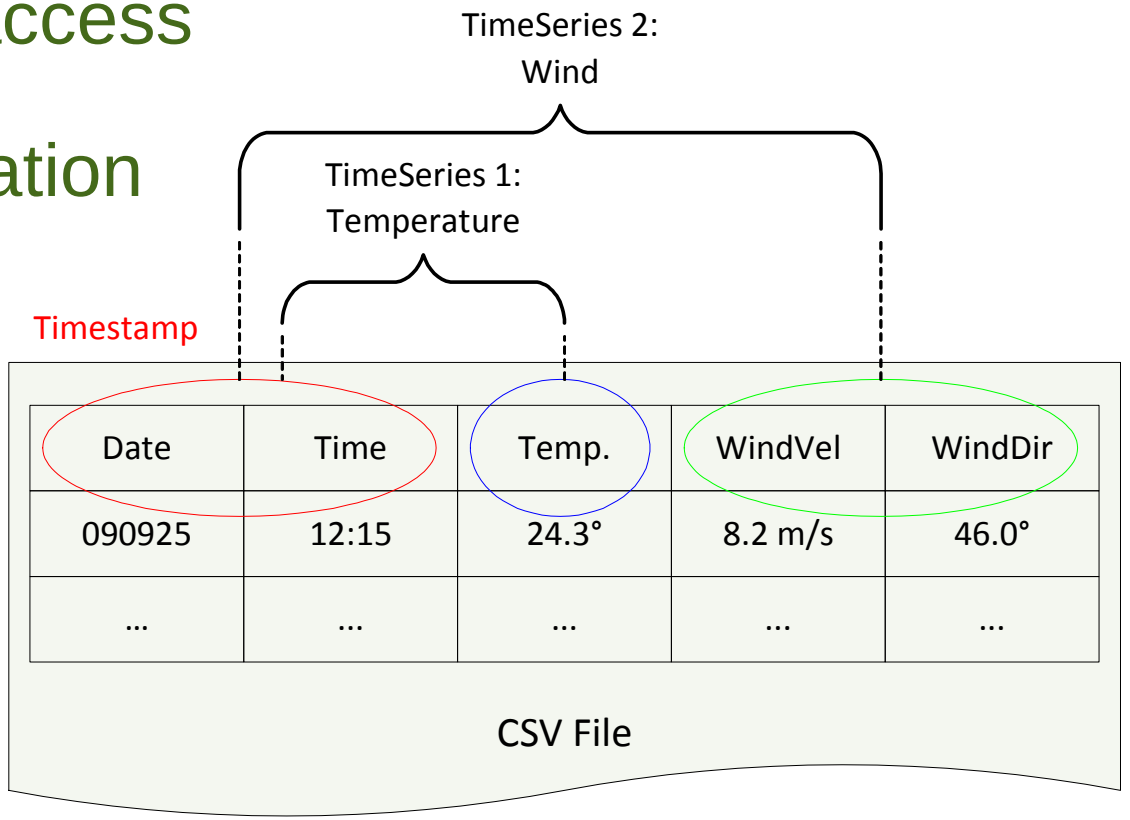
Data connector: SOS DataStore

- Accessing Sensor Observation Services
- Read: Observations, SensorML, Features
- Write: Observations, RegisterSensor
- Flexible parser
  (e.g. SOS 1.0 / 0.31)



Data connector: CSV DataStore

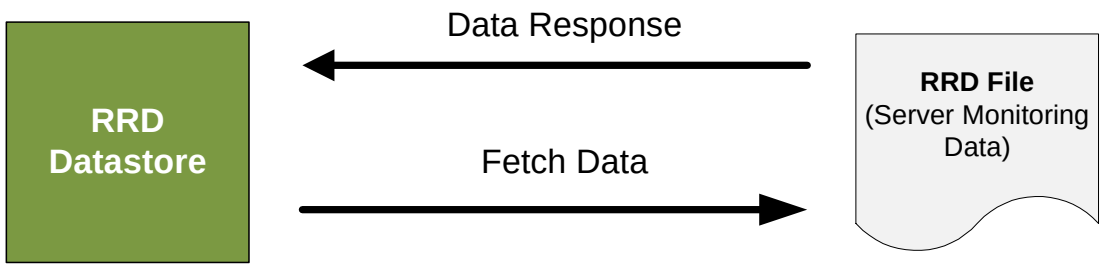
- Access Comma Separated Value Files
- Read and write access
- Flexible configuration
  - Separators
  - Formats
  - Values



Data connector: RRD DataSource

- Round Robin Database (http://oss.oetiker.ch/rrdtool/)
- Currently read-only access to data
- Integration of existing RRD files

  (e.g. from system monitoring applications)



Data connector: AnySen DataSource

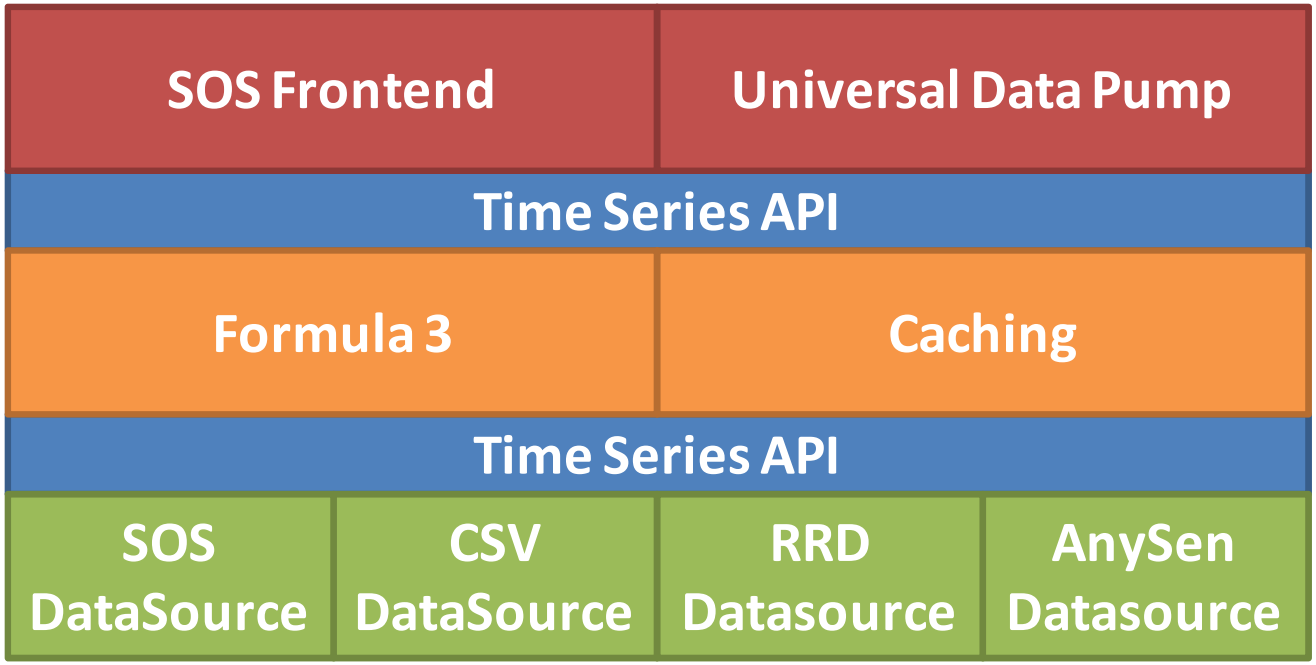
- Data aquisition directly from a sensor
- Designed for serial streaming sensors
- Flexible configuration of parser
- Plug & measure – get config from repository

Data Connector API

```
public interface DataStore {

    // connection
    public void connect(String connectString);
    public void connect(Map<String, String> properties);
    public void disconnect();

    // configuration
    public Object getDataStoreProperty(String key);
    public void setDataStoreProperty(String key, Object value);

    // time series
    public List<String> getTimeSeriesIds();
    public List<String> findTimeSeriesIds(Map<String,
        Object> timeSeriesProperties);

    ...
}
```

Layered Architecture

- Data connector components
- **Core processing components**
- Frontend components

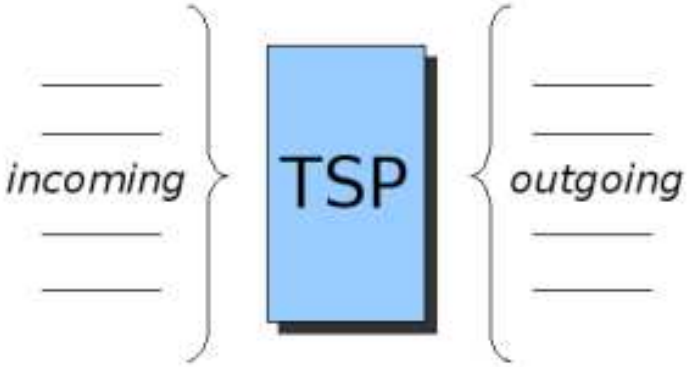


Formula 3

- „Time Series Processor“
- Read time series data
  from connectors via TSAPI
- Process data:

```
< [n] * 2 >
< [n-2 .. n].sum >
< ( t-3 hours .. t ].mean > every 30 mins
```

- Output time series data via TSAPI



Formula 3

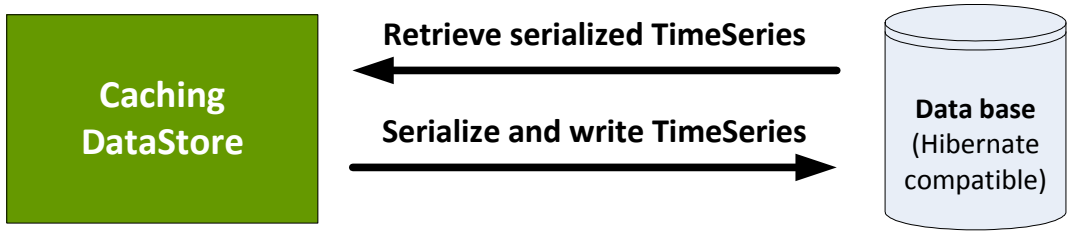
- Functionality
  - Calculation (arithmetic)
  - Parameters for formulas
  - Time patterns, slot/range selection
  - Conditions
  - Predefined functions, user defined functions
- Example use cases
  - Aggregation, Mean values, etc.
  - Filtering, Classification

Formula 3 - implementation

- Design goal: usable as a „standalone“ library on major platforms (Java, DotNET, „native“)
- → Implemented in Python
- TS Toolbox contains Formula 3 running on Jython

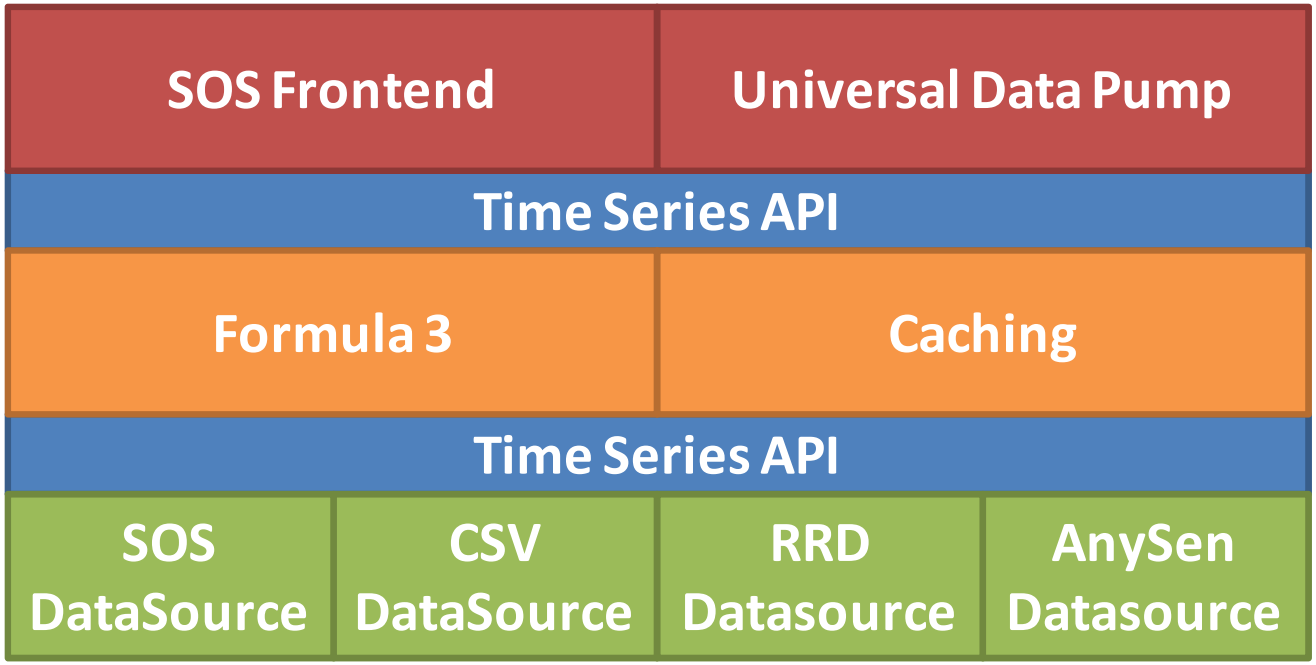
Caching

- Store time series data in local database
- Temporary storage
- Use cases: caching, prefetching, local storage, etc.
- Currently implements configurable prefetching



Layered Architecture

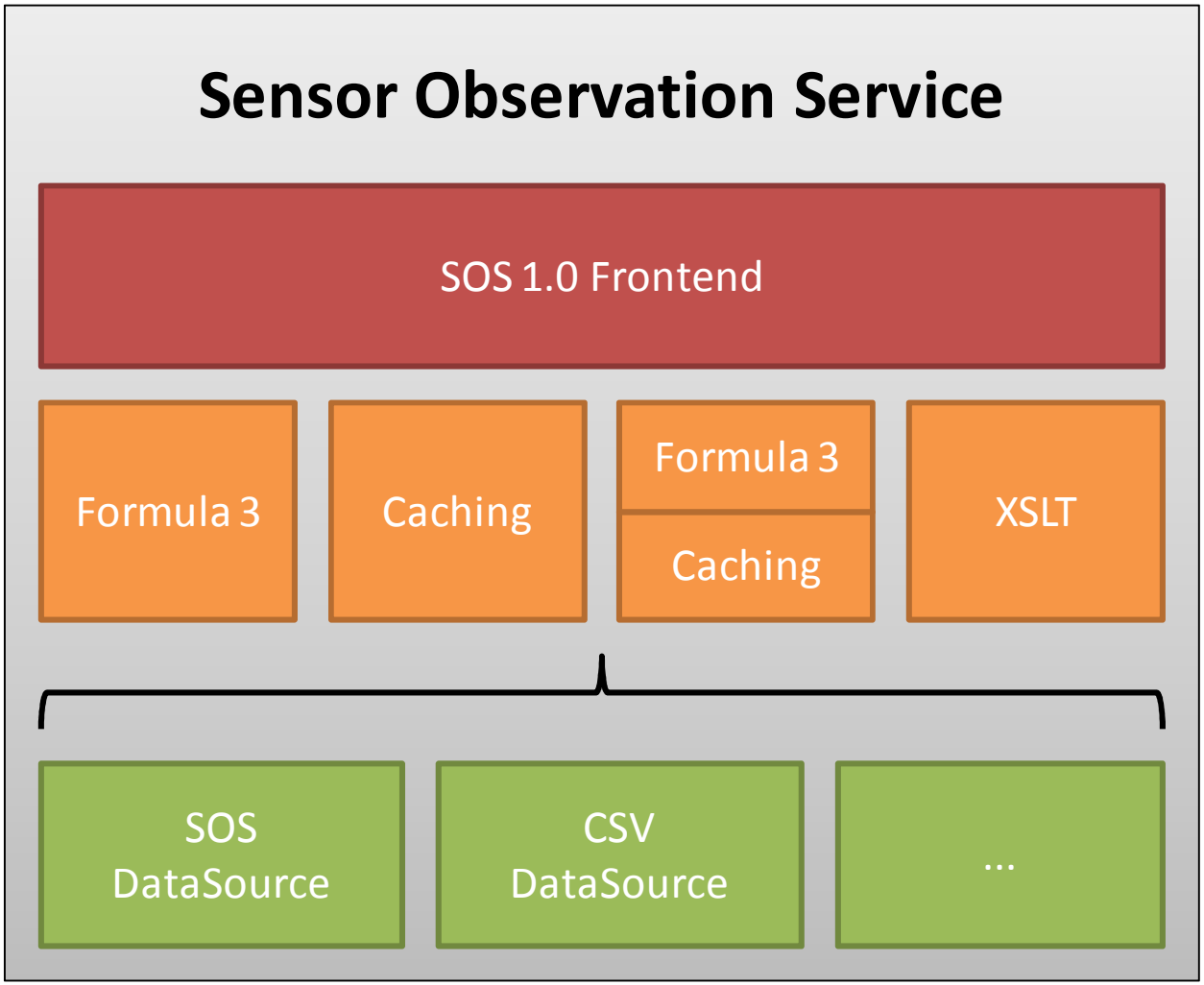
- Data connector components
- Core processing components
- **Frontend components**



SOS Frontend

- Sensor Observation Service Frontend to TSAPI
- Serve data from multiple data connectors
- On-the-fly processing using Formula 3
- Local caching/prefetching of data
- Based on 52°North SOS
- SensorML generation/transformation with XSLT

SOS Frontend: application schema



SOS Frontend: use cases

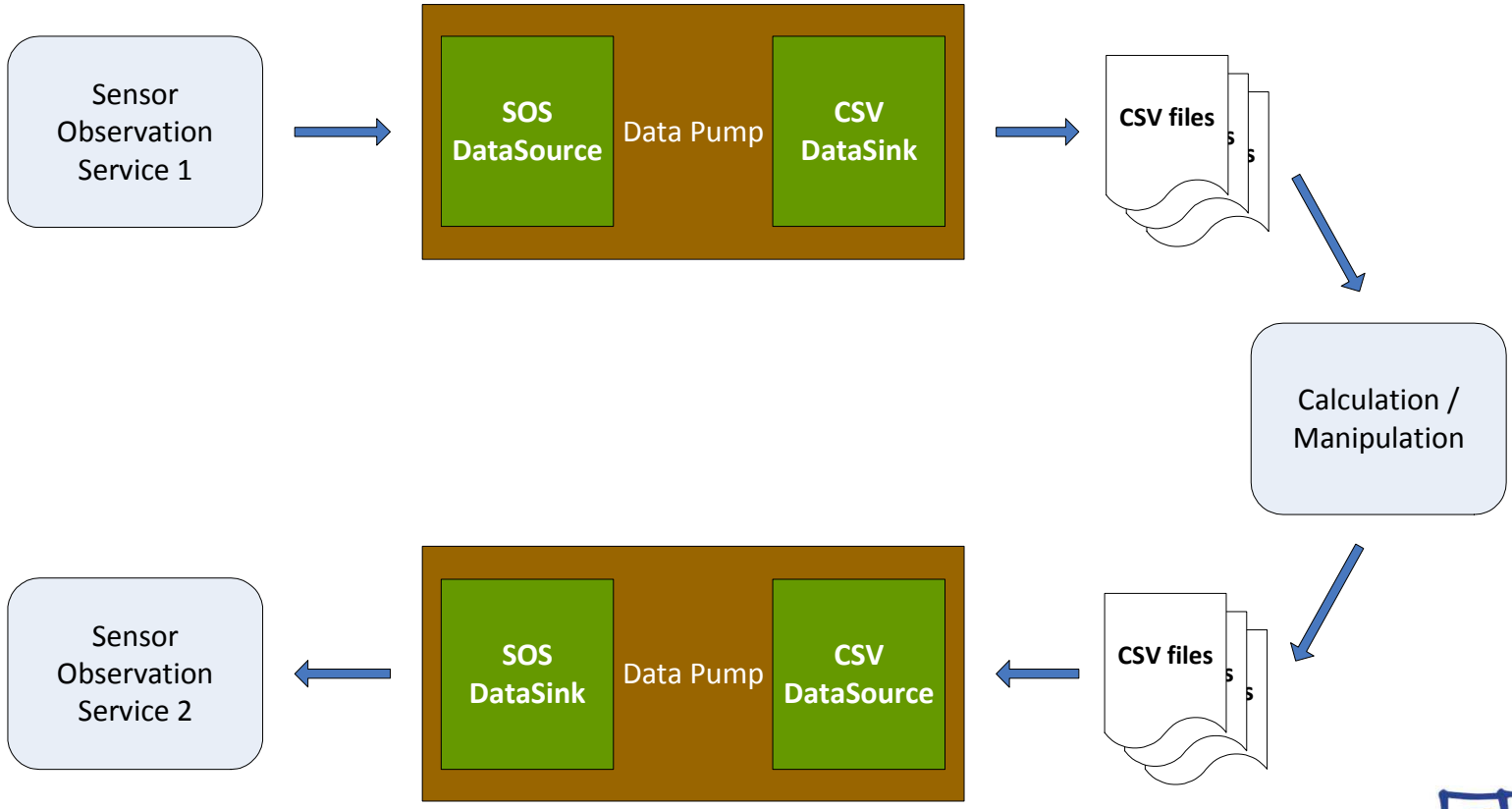
- SOS Interface to DataSource (CSV, Legacy, etc.)
- SOS with on-the-fly processing
- Cascading SOS: custom views, transformations, caching, etc.
- Data preparation for reporting
- …

Universal Data Pump

- Read data from a DataSource
- Write to one or more DataSink(s)
- Configurable (config file and commandline)
- → Batch Processing
- CSV → SOS, SOS → CSV, Legacy → SOS, AnySen → CSV, AnySen → SOS, etc.

Data Pump – Example Scenario

- Export data from SOS into a CSV file
- e.g. for legacy modeling application



Example config file - structure

```
<?xml version="1.0" encoding="UTF-8"?>
<DatapumpConfiguration xmlns="http://enviro.ait.ac.at/datapump/configuration">

  <General>
    <ErrorHandling>
      ...
    </ErrorHandling>
  </General>

  <DataSource>
    <Connection>
      ...
    </Connection>

    <Filter>
      ...
    </Filter>
  </DataSource>

  <DataSink>
    <Connection>
       ...
    </Connection>
  </DataSink>

</DatapumpConfiguration>
```

Example config file - datasource

```
<?xml version="1.0" encoding="UTF-8"?>
<DatapumpConfiguration xmlns="http://enviro.ait.ac.at/datapump/configuration">
  ...
  <DataSource>
    <Connection>
      <Parameter key="ds:protocol" value="SOS"/>
      <Parameter key="ds:connect_string“
                 value="http://enviro5.arcs.ac.at/SOSuwedat10/sos"/>
    </Connection>

    <Filter>
      <Parameter key="ts:offering" value="all"/>
      <Parameter key="ts:observed_property" value="urn:ogc:def:phenomenon:arcs:wiv"/>
      <Parameter key="ts:feature_of_interest" value="urn:ogc:object:feature:arcs:s404"/>

      <!-- Query time interval -->
      <Parameter key="time:start" value="2007-05-01T00:00:00+0200"/>
      <Parameter key="time:end"   value="2007-05-24T01:59:59+0200"/>
    </Filter>
  </DataSource>
  ...
</DatapumpConfiguration>
```

Example config file - datasink

```
<?xml version="1.0" encoding="UTF-8"?>
<DatapumpConfiguration xmlns="http://enviro.ait.ac.at/datapump/configuration">
  ...
  <DataSource>
     ...
  </DataSource>
  <DataSink>
    <Connection>
      <Parameter key="ds:protocol" value="CSV"/>

      <!-- Filename "-" stands for standard output -->
      <Parameter key="ds:connect_string" value="-"/>

      <!-- Column labels -->
      <Parameter key="csv:labels" value="Time|Value|Unit"/>

      <!-- Which properties to put in which column -->
      <Parameter key="csv:fields" value="time|slot:value:value|ts:ts:unit"/>

      <!-- Dateformat for time stamps -->
      <Parameter key="csv:format:dateformat" value="yyyy-MM-dd HH:mm"/>
    </Connection>
  </DataSink>
</DatapumpConfiguration>
```

Datapump running

- Running the following command:

```
$ java -jar datapump.jar config.xml
```

- Generates a file with the following content:

```
Time;Value;Unit
2007-05-01 00:00;1.2346;m/s
2007-05-01 00:30;0.73287;m/s
2007-05-01 01:00;1.0861;m/s
2007-05-01 01:30;0.98261;m/s
2007-05-01 02:00;0.95999;m/s
2007-05-01 02:30;1.1562;m/s
2007-05-01 03:00;1.0709;m/s
2007-05-01 03:30;0.64072;m/s
...
```

Time Series Toolbox Availability

- Developed by AIT within the SANY project
- Available under dual license :
  open source (GPL) and commercial license
- Next version currently being released at:
  http://sourceforge.net/projects/timeseriestool/
- For more info contact:
  thomas.bleier@ait.ac.at

Future developments

- API cleanup / refactoring
- More dataconnectors
- More applications

- <insert your requirements here>

SANY – Sensors Anywhere

SANY is an

Integrated Project

(contract number 0033564)

co-funded by the Information Society and Media DG of the European Commission

within the RTD activities

of the

Thematic Priority Information Society Technologies”

Information Society Technologies

SANY Consortium

Project acronym	SANY
Project reference	IST-2006-033564
Project type	Integrated Project
Start date	01/09/2006
Duration	36 months
Budget	11,2 M€
EC contribution	7,0 M€

S@NY Sensors Anywhere

Thank you!

Questions?

Information Society Technologies