Mensch-Computer-Interaktion
Human-Computer Interaction

US Assignments

(Winter Term 2014/2015):
Assignment III: Sens-ation

Christoph Beckmann
Human-Computer Interaction Group
University of Bamberg
96045 Bamberg, Germany

<firstname>.<lastname>(at)uni-bamberg.de

Introduction

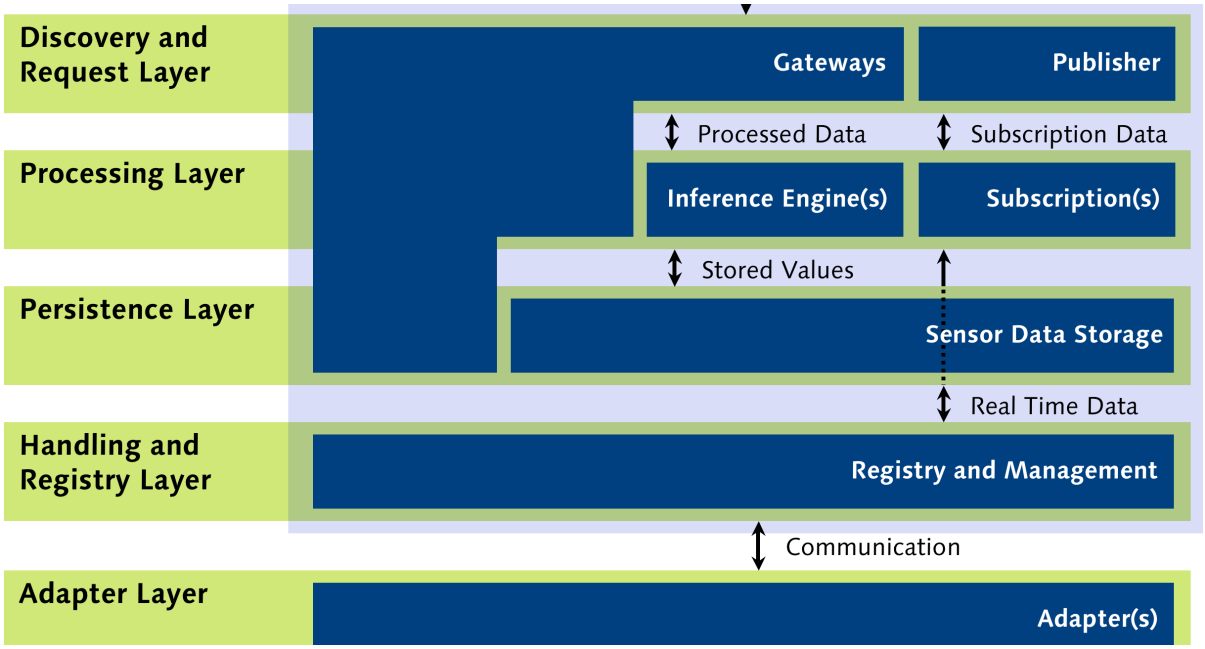
Sensor-based infrastructure for developing ubiquitous environments (service-oriented)

- **Collects** sensor event data
- **Stores** sensor event data
- Allows for **inference** from sensor event data
- **Promote** sensor event data to various clients

Connections via

- Sockets
- **XML-RPC**
- SOAP
- Peer to Peer

Introduction (cont'd)



References

Gross, T., Egla, T. and Marquardt, N. *Sens-ation: A Service-Oriented Platform for Developing Sensor-Based Infrastructures*. International Journal of Internet Protocol Technology (IJIPT) 1, 3 (2006). pp. 159-167. (ISSN Online: 1743-8217, ISSN Print: 1743-8209).

The Sens-ation Platform

Installation
Usage
Location description
Sensor description
Sensor types
Methods provided by Sens-ation via XML-RPC

The Sens-ation Platform (cont'd)

Installation

- Unzip the Sens-ation_1.73.zip archive
- **Configure** the server to your needs
  - `server.properties`
  - `gateway.properties`
- **Start** the server
  - `java -jar sens-ation_1.73.jar`
- **Optionally** install and use
  - A MySQL database (when persistence is required)
  - A Web-frontend (available upon request)

The Sens-ation Platform (cont'd)

Usage

- Watch the **Sens-ation log**
  - Every platform action is documented there
- **CLI** Commands
  - `list { sensortypes | sensors | locations }`
    List the sensor types, sensors, or locations known to the platform
  - `delete { <sensorID> | <locationID> }`
    Delete a sensor or location
  - `value <sensorID>`
    Print the last sensor event value of a sensor
  - `server rc`
    List the subscribers of the platform
  - `exit`
    Shut down Sens-ation

The Sens-ation Platform (cont'd)

Location XML Description

```
<Location id="ID (e.g., WE5/01.045)">
  <Description>A description</Description>
  <DegreeOfLongitude>float</DegreeOfLongitude>
  <DegreeOfLatitude>float</DegreeOfLatitude>
  <HeightAboveSeaLevel>float</HeightAboveSeaLevel>
  <Type>inside, outside, or omit completely</Type>
</Location>
```

The Sens-ation Platform (cont'd)

Sensor XML Description

```
<Sensor id="ID (e.g., ESB1Temp)" class="Sensor type">
  <Description>A description</Description>
  <HardwareID />
  <Command />
  <LocationID>A location name</LocationID>
  <Owner>Your name</Owner>
  <Comment>A usefull comment</Comment>
  <AvailableSince>yyyy-mm-dd hh:mm:ss</AvailableSince>
  <AvailableUntil>yyyy-mm-dd hh:mm:ss</AvailableUntil>
  <SensorActivity activity="active" />
  <NativeDataType>String, Float or Integer</NativeDataType>
  <MinimumValue>Appropriate to sensor and data type</MinimumValue>
  <MaximumValue>Appropriate to sensor and data type</MaximumValue>
</Sensor>
```


The Sens-ation Platform (cont'd)

Sensor types

- **ASCII** (Keyboard ASCII input)
- **Binary** (Sensor with binary state: 0 or 1, true or false)
- **Button** (Hardware Button, true or false)
- **CellPhoneRC** (Use cell phone as remote control)
- **CellPhoneState** (Cell phone state)
- **CellPhoneText** (Cell phone text messages)
- **Infrared** (Infrared command receiver)
- **Light** (Light intensity measurement)
- **MessengerStatus** (Instant messenger availability status)
- **MessengerText** (Instant messenger text message)
- **Movement** (Movement sensor. IR passive)
- **Noise** (Microphone noise measurement)
- **NoiseAverage** (Microphone, average value)
- **NoiseCounter** (Microphone, counter)
- **Other** (Type not specified)
- **Presence** (Presence information, CML PRIMI project)
- **Service** (Calculated service values)
- **Temperature** (Temperature sensor module)
- **Vibration** (Vibration sensor)

The Sens-ation Platform (cont'd)

Sens-ation provides **methods** to

- **Register**
  - Locations
  - Sensors
- **Submit** sensor events
- **Obtain** data about locations, sensors, and events
- **Subscribe** to sensors

SensorPort

Both via XML-RPC

GatewayXMLRPC

The Sens-ation Platform (cont'd)

SensorPort

- **XML-RPC handler** for
  - Checking the **availability** of services
  - **Registration** of
    - New locations
    - New sensors
  - **Notification** about new sensor events from sensor

SensorPort.ping

- `String ping()`
- Checks if the server is running
- Returns a String with a message

The Sens-ation Platform (cont'd)

SensorPort.registerLocation

- `boolean registerLocation(String locationXML)`
- Register **new location** for sensor
- Parameters:
  - locationXML (String containing the XML description of a location)
- Returns boolean (true if the registration was successful)

The Sens-ation Platform (cont'd)

SensorPort.updateSensor

- `String updateSensor(String sensorXML)`
- **Update sensor** information on the sensor platform
- If a sensor **exists the values will be updated**; if not a new sensor will be created
- Parameters:
  - sensorXML (String containing the XML description of a sensor)
- Returns string with the sensor ID if registration was successful

The Sens-ation Platform (cont'd)

SensorPort.notify

- `boolean notify(String sensorID, String datestamp, String event)`
- **Notify a sensor with a new event value**
- Parameters:
  - sensorID (String of the sensorID)
  - datestamp (String of the datestamp of the event ISO 8601 format: yyyy-MM-dd HH:mm:ss, e.g. 2014-11-06 17:33:44)
  - event (String of the event message, contains either raw value or XML descriptions)
- Returns boolean (true if notification succeed)

The Sens-ation Platform (cont'd)

GatewayXMLRPC

- **Client XML-RPC access** to Sens-ation
- Provides **various methods** to obtain data on locations, sensors, and events
- Method's name **enclose what information to obtain** and its data type, e.g.,
  - `Hashmap getValuesHashMap(String)`
  - `Hashtable getValuesHashtable(String)`
  - `String getValueString(String)`
  - `Vector getValuesVector(String)`
  - `String getValuesXML(String)`

The Sens-ation Platform (cont'd)

GatewayXMLRPC.getValueString

- `String getValueString(String sensorID)`
- Returns **the last sensor event value** of a sensor as String
- Parameters:
  - sensorID (String of the sensor ID of the sensor of interest)
- Returns a String containing the last event value

The Sens-ation Platform (cont'd)

Pub/Sub via XML-RPC

- Sens-ation provides a **simple publish-subscribe** mechanism via XML-RPC
  - **Register** for new events of a specific sensor
  - Implementation of an **XML-RPC server on the receiver's side**
  - ```
    String StableXMLRPCClient.notify(String sensorID,
                                     String dateStamp,
                                     String value)
    ```
    String as return type to **handle incoming notifications**
  - When a new sensor event on the specified sensor occurs, the StableXML-RPCClient gets notified via a XML-RPC call
- GatewayXMLRPC provides methods
  - Register and unregister subscribers
    ```

The Sens-ation Platform (cont'd)

GatewayXMLRPC.register

- Method for **subscribing for continual notification** via XMLRPC
  - `String register(String ip, String sensorID, String port)`
- Parameters:
  - ip (IP address of the client as String)
  - sensorID (ID of the sensor the client wants to subscribe to as String)
  - port (the port the client is running on as String)
- Returns a string containing either "done" if the registry was successful or "error" if not

The Sens-ation Platform (cont'd)

GatewayXMLRPC.unregister

- Method for **unregistering from continual** services (XMLRPC)
  - `String unregister(String ip, String sensorID)`
- Parameters:
  - ip (IP address of the client as String)
  - sensorID (ID of the sensor the client wants to subscribe to as String)
- Returns a String containing either "done" if the unregistering was successful or "error" if not

Prerequisites

We use

- **J2SE 7.0** available at http://docs.oracle.com/javase/7/docs/
- **Apache XML-RPC 1.2 b1** (via last assignment)
- **Sens-ation-1.73** (via BSCW)

This assignment does **not** require the use of an integrated development environment (IDE)

You are encouraged **not** to use an IDE in order to understand what you are doing at this level

Preparation

Install, configure, and start your copy of **Sens-ation**

Get familiar with **Sens-ation sensor and client programming**. (refer to these slides and to the supplied javadoc)

Assignment

Sens-ation Sensor & Client

- Write two small **Java programs**
- A **Sens-ation sensor** (e.g., a sensor as implemented in before)
  - **Registers** as a sensor to Sens-ation
  - Reasonably and continuously **delivers values** to Sens-ation
- A **Sens-ation client**
  - **Accesses** the sensor values
  - **Subscribes** to the sensor
  - **Reacts** to the values, e.g., print, make noise

Document your code

- Describe your code in **javadoc comments**

Assignment (cont'd)

Implement your code using the build environment

Use the following package name in your Java file:

- `package de.cmlab.ubicomp;`

Compile and create the **documentation** of your code

- Adapt the ant build file
- Use ant to
  - Build your code
  - Create the javadocs of your code

Submit your **complete build environment**

Submission

This assignment is due on **19 November 2014**

Please submit your results according to the **submission guidelines** as archives in .tar.gz or .zip file format

Credits

Two (2) credits can be achieved in this assignment

- Sens-ation sensor program 40%
- Sens-ation client program 40%
- Java docs 10%
- Build environment adaptation (including ant file) 10%

Comments & Questions

Feel free to send questions and comments about this assignment to

christoph.beckmann@uni-bamberg.de

To process your email quickly and due to SPAM-filters, please use the following subject line

HCI-US/<your concern>

Mensch-Computer-Interaktion
Human-Computer Interaction

Thank you!