Research Project:

Collaboration Bus

Nicolai Marquardt
Computer-Supported Cooperative Work Group
Faculty of Media
Bauhaus-University Weimar
Bauhausstr. 11, 99423 Weimar, Germany

<firstname>.<lastname>(at)medien.uni-weimar.de

Outline

- Introduction
- Architecture
- Implementation Details
- Graphical User Interface
- Application Scenarios
- DEMO
- Conclusion and Future Work

Introduction

- “Software tool for the end-user to easily create new envisioned sensor-based applications”

- Universal combination of sensors and actuators, and the specification of interpretation and filter settings
- Connected to the Sens-ation sensor infrastructure

Introduction (cont'd)

Objectives (derived from the related work research)

1. Focus especially on pipeline compositions to support and enhance work and everyday life
2. Intuitive, easy-to-learn interface, with specialized functions (not too generic)
3. Hiding the graph theory as much as possible
4. Sharing mechanism, exchange of compositions
5. The „repository" view, personal control interface
6. Templates and patterns
7. No replacement for the complex „inference engine" modules
8. Covers Ubiquitous Computing and CSCW applications

Introduction (cont'd)

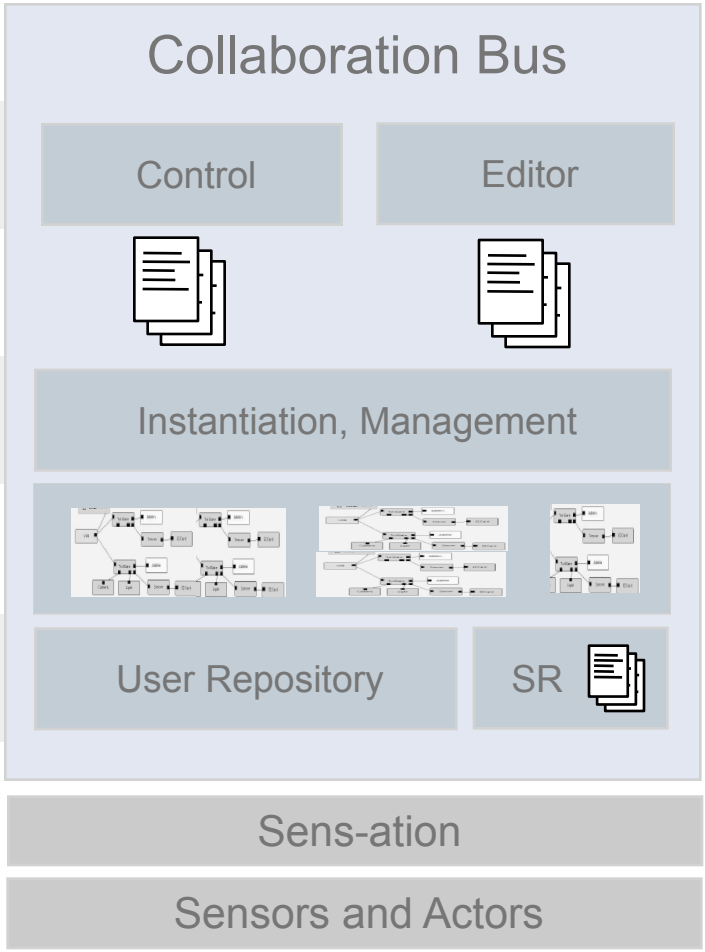
Presentation layer: editor and personal repository control interface

Data exchange: description of current data flows and parameters

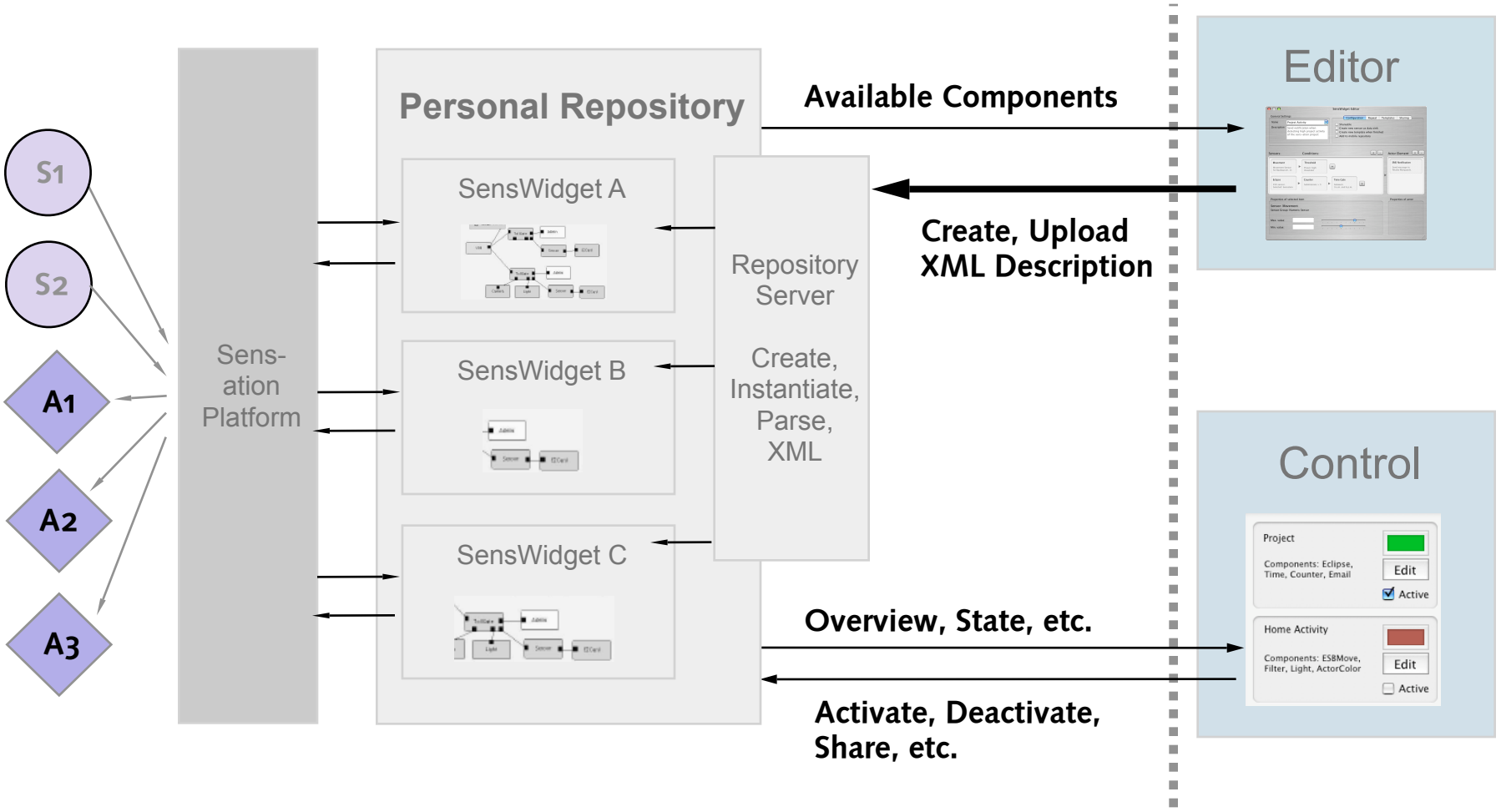
Instance layer: creating running instances of the pipeline compositions

Personal Repository: individual SensWidgets, running threads

Global repository: storage for user repository, shared repository, etc.



Architecture



Architecture (cont'd)

- Main software components of Collaboration Bus
  - Classes for the user repository and pipeline compositions
  - Control GUI classes
  - Editor GUI, view/controller for the component model
  - Components: sources, filter and actuators and the abstract base class
  - Visualization classes
  - Remote repository server
  - Shared repository
  - Utility classes

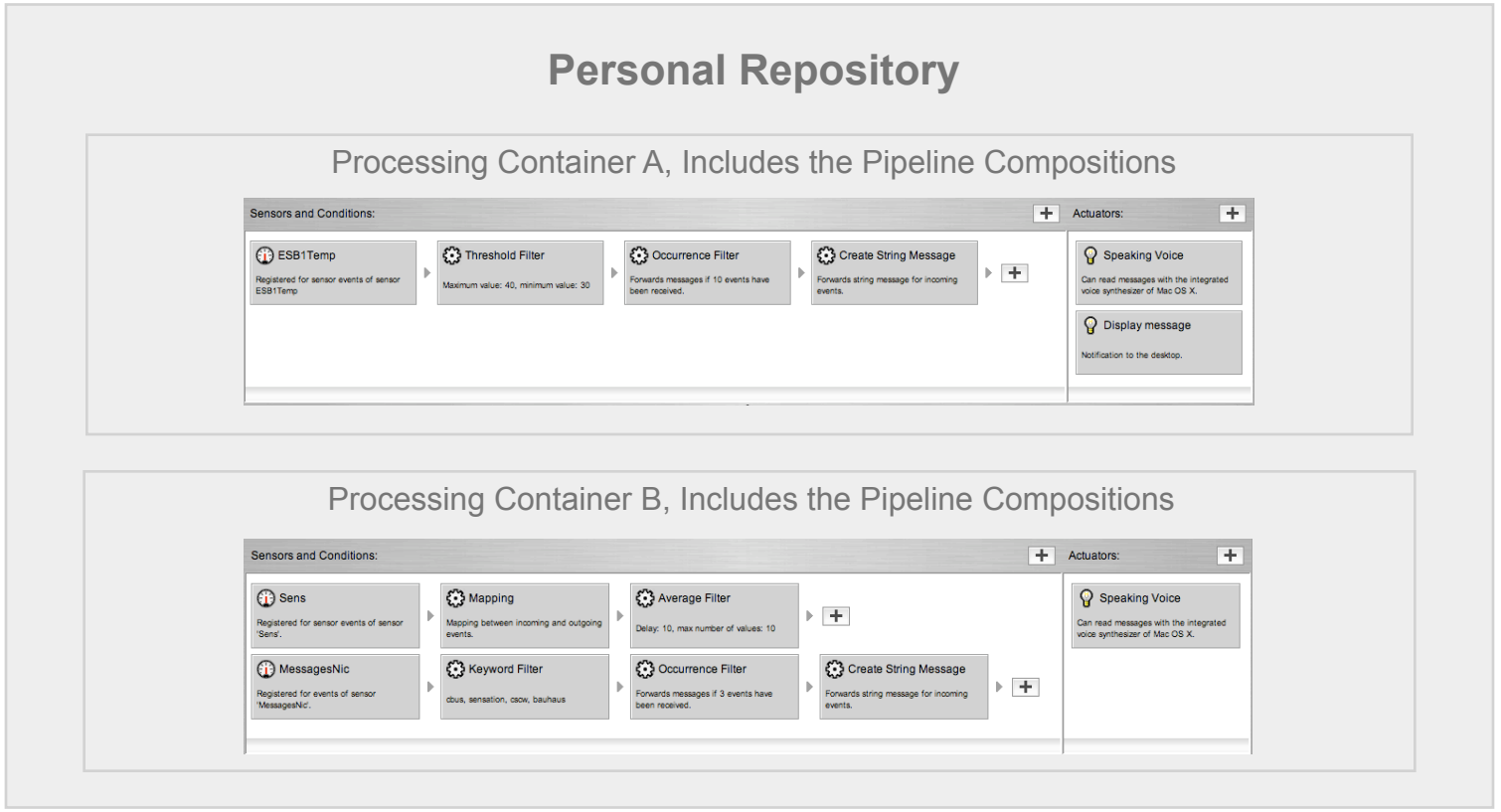
Implementation Details

- Personal repository and processing container
  - Personal repository is the main class to manage all processing containers of a user
    - Methods to add and remove containers
    - Clone methods and sharing of processing containers
  - The processing container contains the pipeline composition
    - Can start the component threads for pipeline processing
    - Registration of global observers (visualizations)
    - Establishes all pipeline connections between components
    - Handles all components: sources, filter and actuators/sinks

Implementation Details (cont'd)

- Components and pipes
  - Processing of the incoming data values of the sensor sources
  - All pipeline components derived from the `AbstractComponent` class
  - Implement `run()`, `start()`, `stop()` method for thread execution
  - Common methods: `notify()`, `forward()`, `reset()`, `init()`
  - Each component has its own event queue
  - Pipelines are created dynamically of the processing container before pipeline execution starts; connect all components
  - Each component divided in model class and view/controller class
  - View wrappers provide multiple visualizations: complete container, components, preference panel

Implementation Details (cont'd)



Implementation Details (cont'd)

- Filter components
  - Processing of the incoming data values of the sensor sources
  - Implementations:
    - Threshold filter: specify lower and upper limit
    - Keyword filter: search for keywords in a string event
    - Average filter: calculates the average value of numeric events
    - Gate timer: pass values every x seconds
    - Occurrence filter: save values before forwarding
    - Generate string: creates string message, with placeholders
    - Mapping table: universal translation of events
  - Filters can be arranged in any order

Implementation Details (cont'd)

- Actuator components
  - Sink components of the pipeline compositions
  - Actions for hardware (e.g. relay board) and software (e.g. desktop notification)
  - Implementations:
    - RSS feed: writes entries to RSS file, using the RSS4J [ChurchillObjects 2005] library to edit and create entries
    - Relay board: can control the 8 ports of the relay board
    - Desktop notification: using the Growl [Forsythe 2005] OS X notification system, Java binding class
    - Color panel: can display 3 color states
    - SMS Gateway: forward to an SMS gateway (control window)
    - Start application: can start any OS X application
    - Sound control: mute the sound speakers of the system
    - Speech: read out messages with the OS X speech synthesizer

Implementation Details (cont'd)

- Technology aspects
  - Object serialization:
    - XML serialization of seven object types (e.g. personal repository, processing container) with XStream [Codehaus 2005b] library
    - Using alias names for objects
    - `XMLUtility` method provides `toXML()` and `fromXML()` method for object instantiation/serialization
    - The serialized repository XML data can be transfered from the `Control` software to the remote repository server (and vice versa)
    - Nested composition of the XML hierarchy

Implementation Details (cont'd)

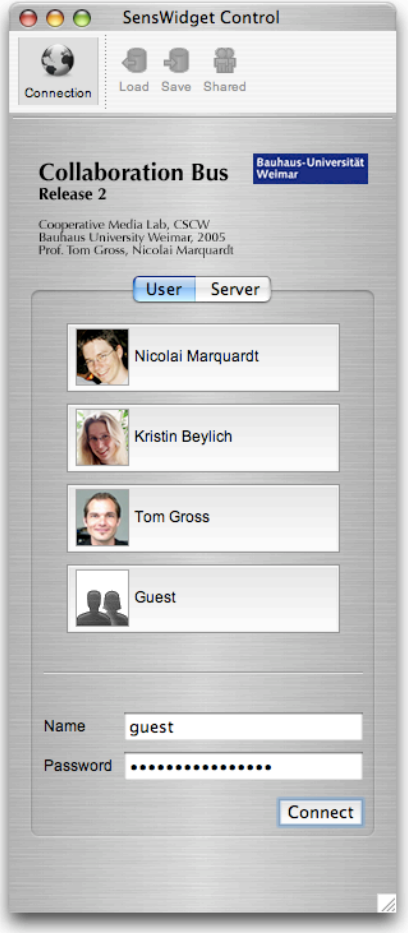
- Technology aspects
  - XML parsing:
    - Parsing the XML documents into JDOM [Hunter & McLaughlin 2005] objects, using the Xerces parser [The Apache Foundation 2005]
    - Node selection of the document with XPath
    - Using the Jaxen [Codehaus 2005a] library and XOM [Harold 2005] to create XPath expressions
      - `XPath.selectNodes(jdomDocument, "/Sensors//Sensor");`
      - `XPath.selectNodes(jdomDocument,"/Sensors//Sensor [@LocationID='" + location + "']");`
    - Useful for exploration of the Sens-ation sensor list XML data (all available sensors and their sensor description)

Graphical User Interface

- Implementation of the user interface in Java Swing
- Optimized GUI components for Mac OS X with the Quaqua library [Randelshofer 2005]
  - JBrowser: tree model based view with lists expanding from left to right (like the Finder)
  - Improved Apple look-and-feel : striped view, changed dialogs, interface elements (toolbar, buttons)
- Helper class `GUIFactory`
- View representations divided into many subclasses, responsible for the view of certain model objects

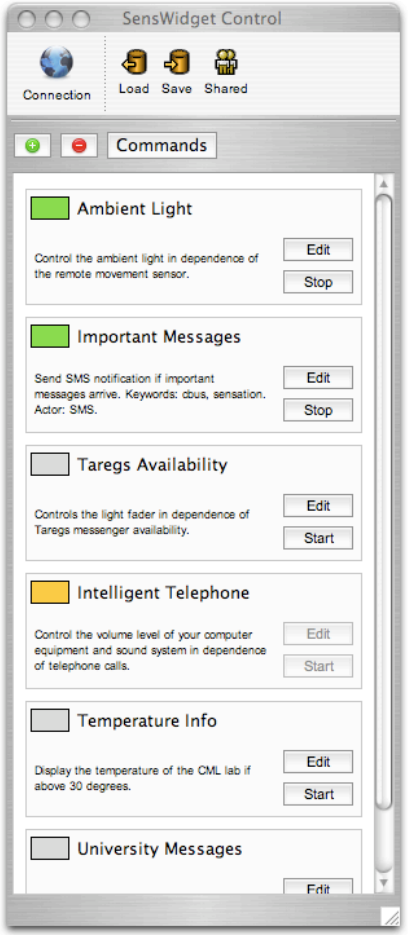
Graphical User Interface (cont'd)

- Control
  - **Login dialog**
  - Server properties: Sens-ation instance, repository server
  - Opens communication channels, loads user repository, requests Sens-ation information of available sensors



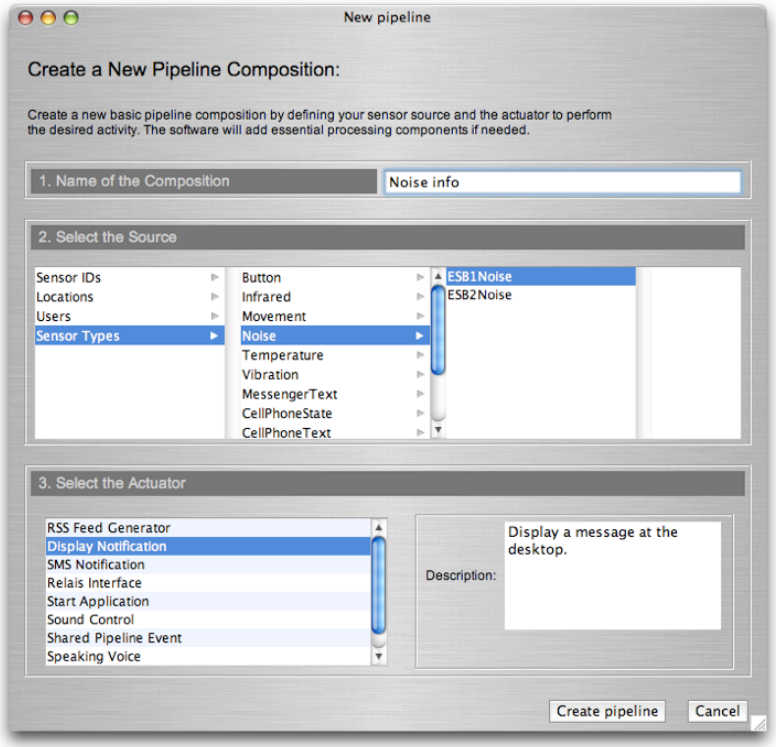
Graphical User Interface (cont'd)

- Control
  - **Login dialog**
  - Server properties: Sens-ation instance, repository server
  - Opens communication channels, loads user repository, requests information of available sensors from Sens-ation
  - **Personal repository view**
  - Start, stop, edit, clone, quick share
  - Access remote repository: load, save, shared repository



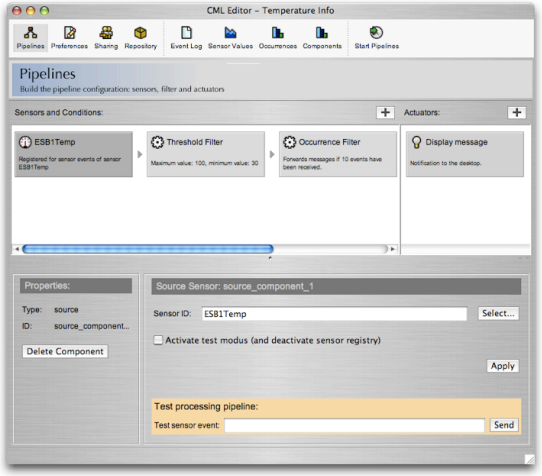
Graphical User Interface (cont'd)

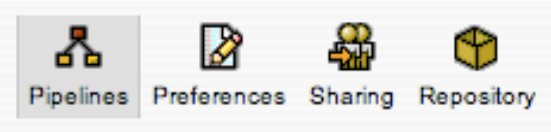
- Control
  - Add new pipeline compositions
  - Assistant dialog
  - Three parts: name, select sensor source, select actuator
  - Available sensors categorized
    - Locations
    - Owners
    - Sensor types
    - Sensor ID (simple)
  - Creates a complete basic pipeline composition

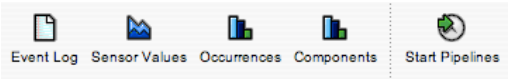


Graphical User Interface (cont'd)

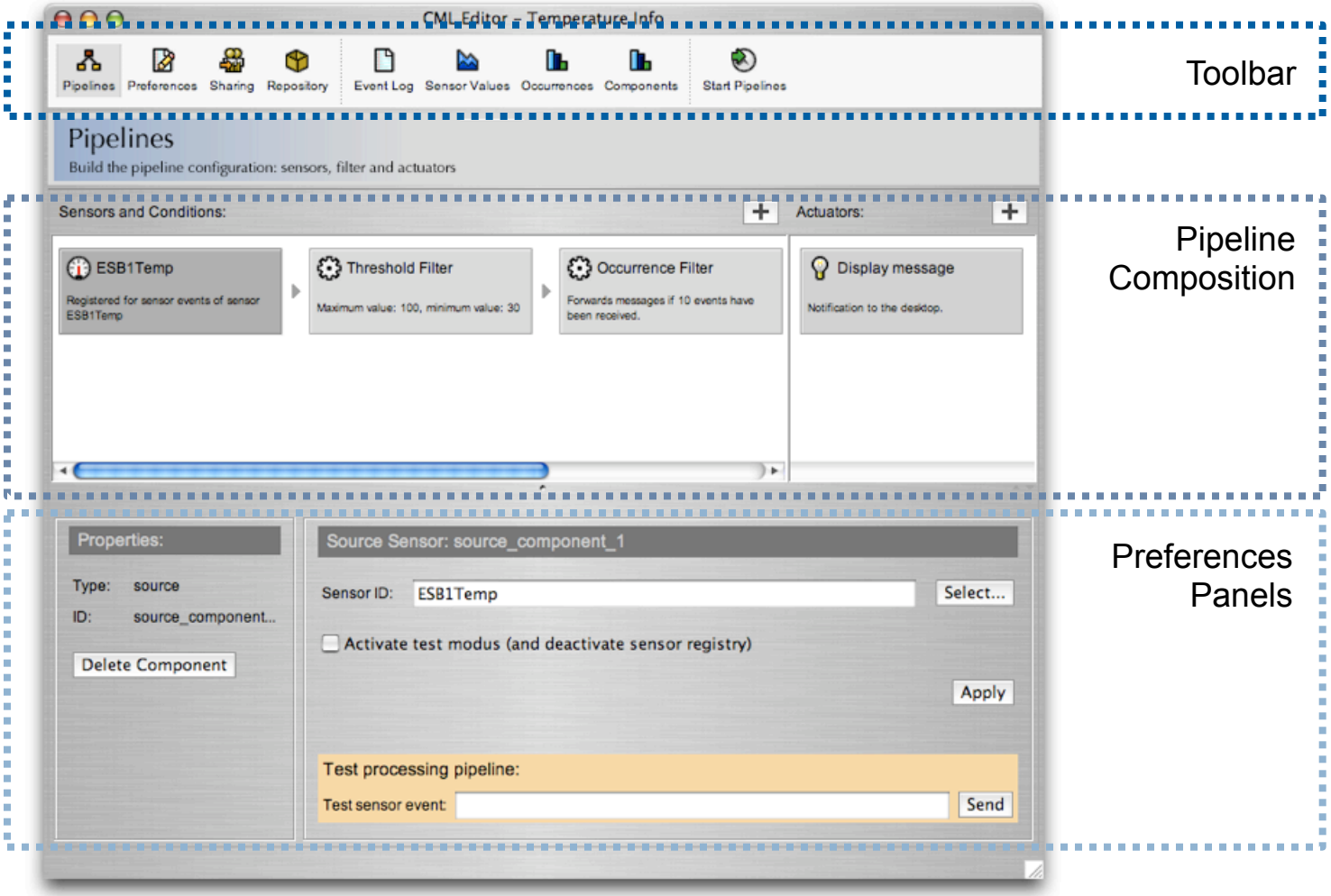
- Editor
  - Four views: pipelines, configuration, sharing, repository source
  - Modify the pipeline composition
  - Add new pipelines
  - Changes component settings: GUI wrapper for each of the pipeline components (View/Controller)
  - Add filter components
  - Add actuators
  - Test pipeline composition in real-time
  - Display event log of all components
  - Show visualization windows







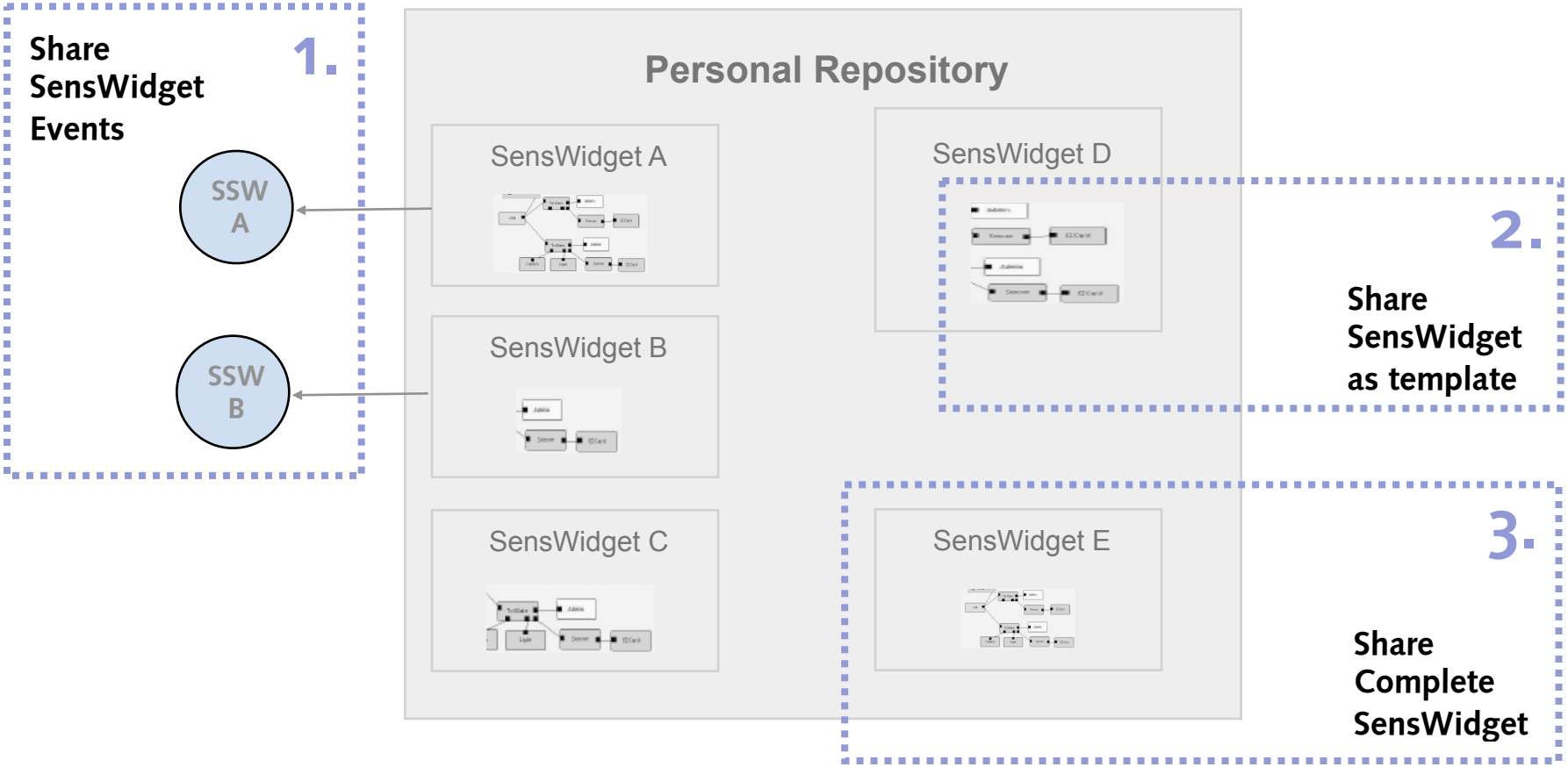
Graphical User Interface (cont'd)



Graphical User Interface (cont'd)

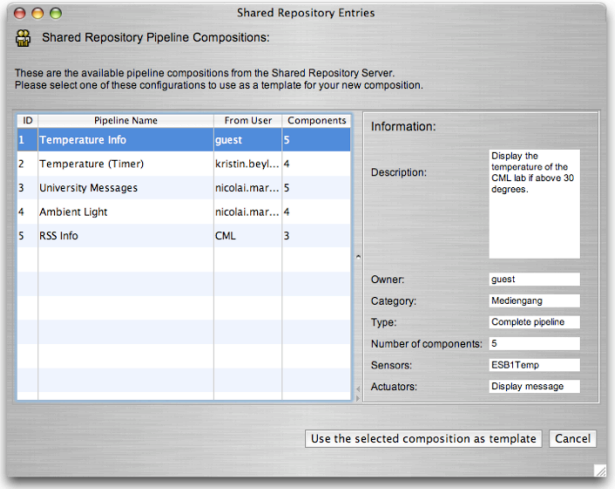
- Visualization
  - Insight into pipeline processes
  - Display occurrences of events and of event values (numeric)
  - Implemented observer structures
  - Using the JFreeChart library [JFree 2005a] [JFree 2005b]
  - Three types of visualization:
    - Time plot of the values
    - Bar chart: the forwarded events of each component
    - Bar chart: the last 60 seconds (in 10 second time slots) of the events of each component

Collaborative Sharing

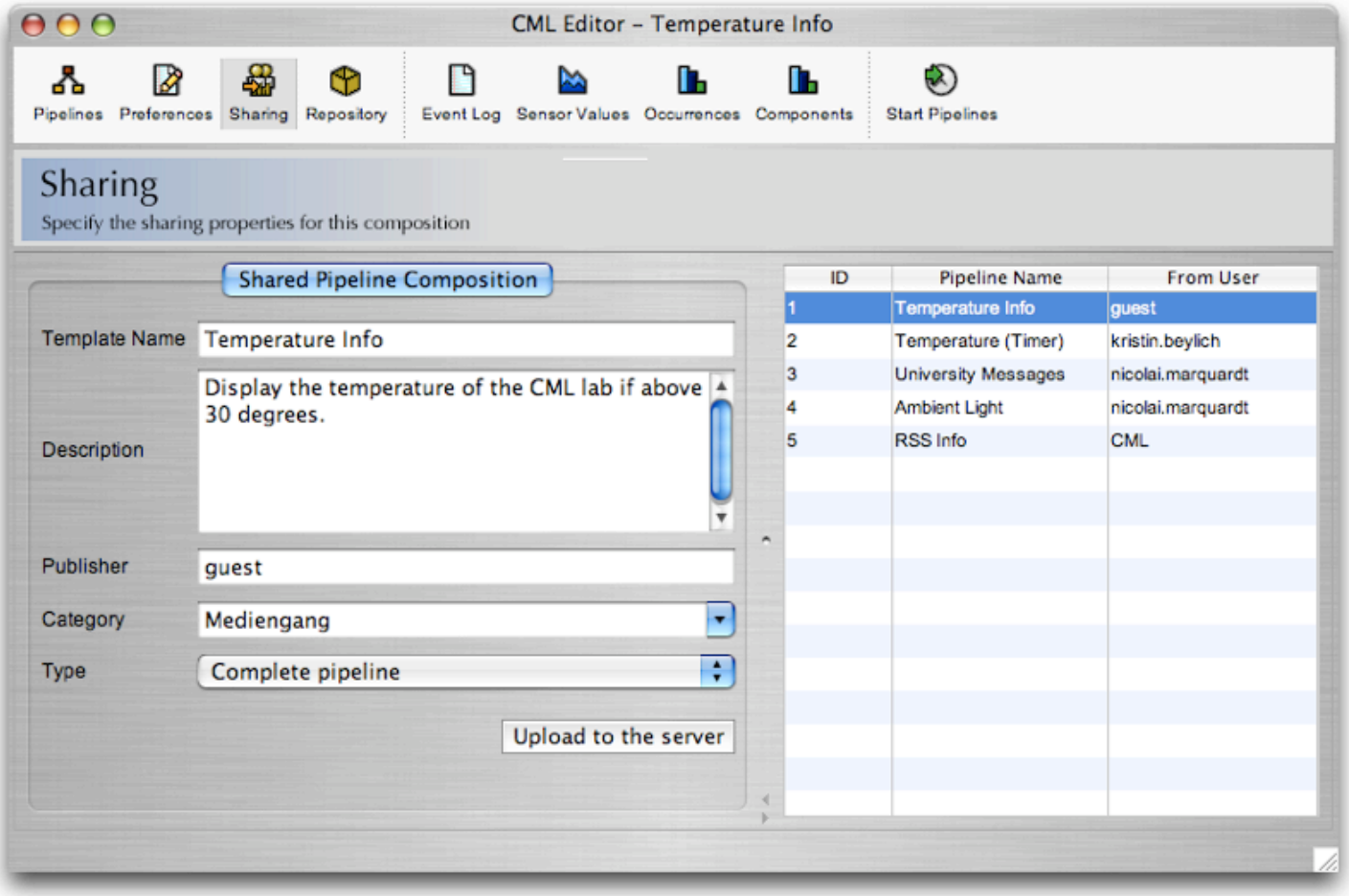


Collaborative Sharing (cont'd)

- Collaborative Sharing
  - Implemented in the control and editor user interface
  - Shared repository entries
  - Handling at the server
  - Three types:
    - Complete composition
    - Abstract template
    - Shared pipeline event (as sensor value)
  - Shared repository exploration: control and editor interface
  - Dynamic instantiation of the shared entries and modification



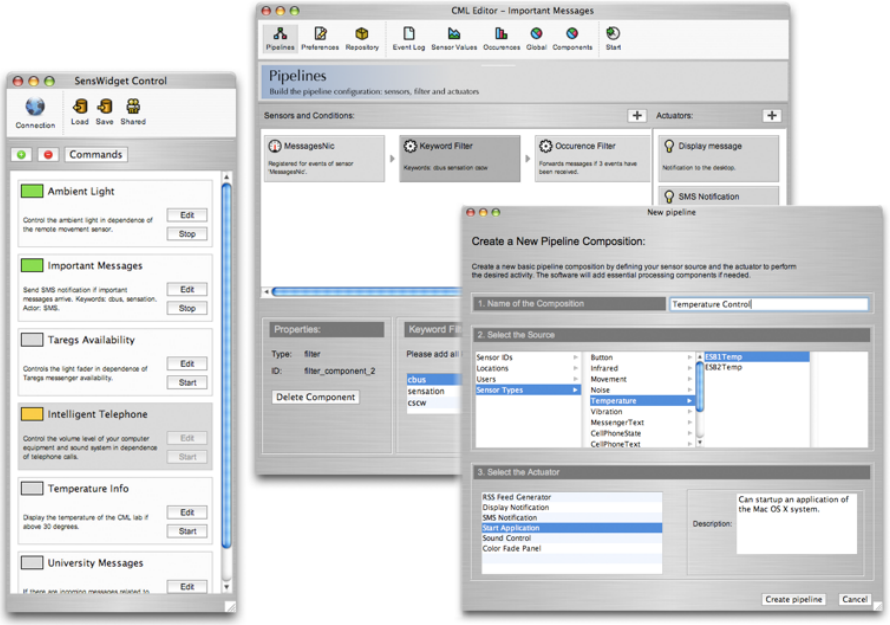
Collaborative Sharing (cont'd)



Application Scenarios

- “Mute all audio sources and the computer sound during telephone calls”
- “Display various information of the state of the working group via the RSS feed screensaver visualization”
- “Notify me if at least five of my friends are available in the instant messenger and activate the messenger software”
- “If an email of the CML members arrive, please read out the message headline”
- “Turn off the light sources if there was no movement for a longer period of time”
- “Create an ambient display at my desktop for the activity at home”
- “Each time I activate the television dim the light sources around”
- “If three important email messages arrive, please send me a SMS notification”
- “If the temperature at home is below 15 degrees, and it is after 6 p.m. please activate the heating and send me notification”

Demo



Conclusion

- Implementation of an easy-to-use editor and control interface for the user
- Reusable components, templates, abstraction
- Collaborative sharing
- Visualization to support the pipeline composition
- Flexible underlying technology layer: XML, dynamic component composition
- Repository server: instance local or remote located

Future Work

- Evaluation of shared pipeline compositions
- Algorithms to provide suggestions for “reasonable” compositions (derived from the shared repository)
- Graphical mapping user interface
- Other filter and actuator components
- Abstract actuator description and universal registry (at the Sens-ation server)
- Coupling with the user authentication algorithm of the Sens-ation platform

References

- The Codehaus Open-Source Project Repository: Jaxen – Xpath Engine for Java, http://www.jaxen.org/, Version 1.1 Beta (Accessed 10/07/05).
- The Codehaus Open-Source Project Repository: XStream, Java Library to Serialze Objects to XML and Back Again, http://xstream.codehaus.org/, Version 1.1.2 (Accessed 10/07/05).
- Hunter, J. and McLaughlin, B. JDOM – Java XML API, http://www.jdom.org/, Version 1.0 (Accessed 10/07/05).
- Randelshofer, W. Quaqua Look and Feel, Mac OS X Enhancement Library, http://www.randelshofer.ch/quaqua/download.html, Version 3.1.1 (Accessed 10/07/05).
- Randelshofer, W. Quaqua Look and Feel – Documentation, http://www.randelshofer.ch/quaqua/ guide/index.html (Manual) and http://www.randelshofer.ch/quaqua/javadoc/index.html (JavaDoc) (Accessed 10/07/05).
- JFree.org Project: JCommon, Class Library for Java, http://www.jfree.org/jcommon/ (Download), http://www.jfree.org/jcommon/jcommon-0.8.9.pdf (Documentation), Version 1.0.0 rc1 (Accessed 10/07/05).
- Forsythe, C. et al.: Growl – Global Notification System for Mac OS X, http://growl.info/documentation/developer/java/ (JavaDoc for the Java binding class) (Accessed 10/07/05).
- ChurchillObjects.com: RSS4J, Java Classes to Create and Parse RSS XML Files, http://www.churchillobjects.com/c/13005.html, Version 0.91 (Accessed 10/07/05).
- The Apache Software Foundation: Xerces, XML Parsing Library for Java, http://xml.apache.org/xerces2-j/, Version 2.7.0 (Accessed 10/07/05).
- The Apache Software Foundation: XML-RPC API for Java, http://ws.apache.org/xmlrpc/, Version 1.1 (Accessed 10/07/05).
- Harold, E. R. XOM, Tree-based API for Processing XML Files in Java, http://www.cafeconleche.org/XOM/, Version 1.0 (Accessed 10/07/05).

Research Project:

Collaboration Bus

Thank you for your attention!