# COLDEX

**Collaborative Learning and Distributed Experimentation**

**Information Society Technologies Programme**
**Project Number: IST-2001-32327**

**Work Report Biodiversity Scenario**

| | |
|---|---|
| **Actual Date of Delivery:** | M26 |
| **Version:** | Final |
| **Work Package:** | Technical report |
| **Partner:** | UDE |
| **Contributing Partner:** | VXU |
| **Authors:** | Jansen, Marc; Gottdenker, Joshua |
| **Contact:** | jansen@collide.info |

## 1. Experiment components

The system developed has several different components. The primary development of this workshop is a server software component that runs on a PC. The server component integrates software client components like a web based inquiry tool and Cool Modes and a hardware component which is basically a device capable of controlling several motors and sensors in order to take control of a physical biotube.

The following is an overview of the Components that are further described in this document:

**Component Overview**

Device Components:
- RCX – 3 sensor inputs, 3 sensor outputs
- Other devices such as the TI – CBL and Leonardo could be integrated later

Server Components:
- Experiment controller – interfaces with the RCX via infrared to implement control and data logging functions which are configured via a webservice
- WebService – receives experiment plans in XML format and delivers them to the demon
- DataBase – stores sensor data and inquiry tool data

Client Components:
- Inquiry tool – provides a graphical interface to chart data from the devices and to build experiment journal artifacts around that data
- CoolModes plug-in – provides a graphical interface to the WebService to build experiment plans and send them to the experiment controller

The major idea of the whole system is to be able to control the biotube remotely, and therefore to allow students to do remote experiments with the biotube. In order to allow the students to describe these experiment we created a Cool Modes reference frame, with the help of which, the students are able to construct the setting of an experiment visually. At the current state of the implementation there are different things that could be defined for an experiment:

- Motors
  - Dynamic
  - Timed
- Sensors
- Data loggings

There are two different kinds of motors that are necessary to provide an experiment in the way it is needed for the biotube. On the one hand there are motors that are activated in regular sequences so that the student can e.g. define patterns for the motors like a motor should be activated for x minutes each y hours. These kind of motors we call timed motors and they are usually used fullfill regular exercises like to give water to the plants in the biotube in regular intervals. On the other hand there are the dynamic motors which behaviour is to some extend related to sensor values. With the help of these dynamic motors it is possible to acomplish patterns like a motor should be activated or deactived if a  sensor value is above or beneath a certain threshold. This basically allows to ensure certain parameters within the biotube since the students can define that e.g. a van should be activated if the temperature is above a certain threshold and the van should again be deactivated if the temperaturs is below another threshold.

Additionally to the motors, sensor configurations could be stored within the description of an experiment. These sensor configurations are used to allow to set certain values relevant to a sensor, e.g. a name could be set for a sensor or the unit in which the sensor reports its data back. Additionally, since the usage of very different sensors is necessary is for the biotube scenario, a possibility is provided to apply a certain mathematical function to the values that the sensors reports. This mathematical function is also part of the sensor description.

Furthermore, within the experiment it is possible to describe the way the data reported by the sensors is logged in a database. Hereby, not only the database could be identified to which the data should be logged, but also the way the amount of data that should be logged over a certain time could be defined. For example, a user could define within its experiment that data from a should be stored every x minutes, and each time not the current value should be stored but the average of n values should be stored. Defining it like this leads the logging daemon to periodically take the sensor values until it has the right amount of data in the right time and then it writes the average of this values to the database and starts over again.

## 2. The infrastructure

Basically, the current infrastructure for the biotube scenario integrates a number of software and hardware components. The relationship between these components is shown in figure 1:

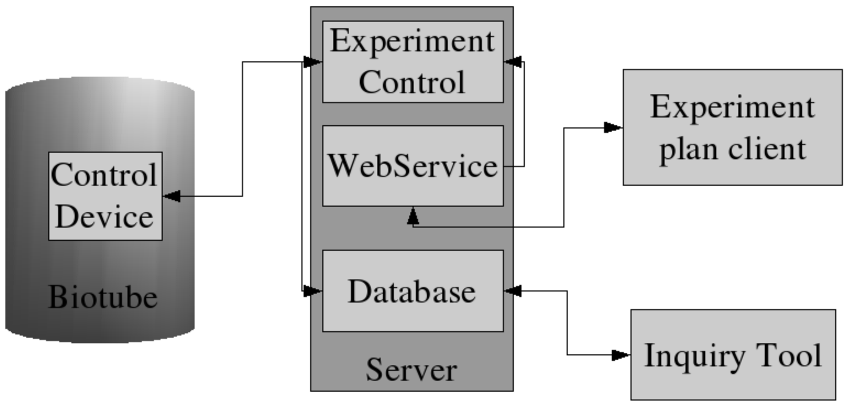


**Figure 1: The layered architecture of the biotube infrastructure**

Within this Figure the different layers could easily be determined. The Biotube, with its control device, is located on the left hand side of the diagram. At this point in time the control device is an RCX with the Lejos environment equipped on it. Nevertheless, the framework foresees that the RCX could be replaced very easily by any other device that is capable of controlling a certain amount of motors and sensors. This is achieved by the fact that the experiment controller, which actually runs on the server layer and is meant to control the ongoing experiment, does not communicate with the controlling device directly but through an interface. Therefore, the only change that is needed to exchange the RCX with  another device is the implementation of the same interface during the implementation of the device itself. The integration of the none-standard devices, like the RCX in this case is done with the help of the framework especially designed to integrate none-standard devices in Java applications.

The WebService is running inside the application server in order to allow the students to send their documents through the WebService to the experiment control server. In this case there is a WebService used to provide a common interface, that is usable for quite a variety of different clients. Additionally, the experiment control server periodically writes sensor values to the database according to its data logging definitions. Also the database access is supported by the "DatabaseDatahandle" component of the framework for the integration of none-standard devices in Java applications. The WebService, deployed in the application server running on the server layer, provides a method that is capable of changing the experiment in the experiment control server by an inter-process communication based on Remote Method Invocation (RMI). There is actually a need for RMI in this case, since from Java Virtual Machine (JVM) point of view, the experiment control server is running remotely in comparison to the application server.

On the client layer there are several different clients possible. Currently, a Cool Modes reference frame is implemented that allows for the visual construction of experiment sets. Furthermore, since the interface for the uploading of an experiment is a WebService a full bunch of different tools could be imagined. Also the inquiry tool, flash and web based, could be extended to define the experiments and later-on upload them to the experiment control server. The initial exploration of using Flash MX to communicate with the WebService was successful.

## 3. The XML description of an experiment plan

The XML description of an experiment setting is based on a Document Type Definition (DTD). The definition of this DTD allows for defining all the related parts of an experiment description.

```
<!ELEMENT Experiment (Motor*, DataLogging*, Configurations)>

<!ELEMENT Motor (Dynamic|Timed)*>
<!ATTLIST Motor motor CDATA #REQUIRED>

<!ELEMENT Dynamic EMPTY>
<!ATTLIST Dynamic sensor      CDATA                #REQUIRED
                  startAction CDATA                #REQUIRED
                  stopAction  CDATA                #REQUIRED
                  action      (activate|passivate) #REQUIRED>

<!ELEMENT Timed EMPTY>
<!ATTLIST Timed each CDATA #REQUIRED
                for  CDATA #REQUIRED>
```

```
<!ELEMENT DataLogging (DatabaseConnection)>
<!ATTLIST DataLogging each           CDATA #REQUIRED
                      numberOfValues CDATA #REQUIRED
                      sensor         CDATA #REQUIRED>

<!ELEMENT DatabaseConnection EMPTY>
<!ATTLIST DatabaseConnection host     CDATA #REQUIRED
                             user     CDATA #REQUIRED
                             pass     CDATA #REQUIRED
                             driver   CDATA #REQUIRED
                             protocol CDATA #REQUIRED
                             database CDATA #REQUIRED>

<!ELEMENT Configurations (MotorConfig*, SensorConfig*)>

<!ELEMENT MotorConfig EMPTY>
<!ATTLIST MotorConfig name  CDATA #IMPLIED
                      speed CDATA #REQUIRED
                      motor CDATA #REQUIRED>

<!ELEMENT SensorConfig EMPTY>
<!ATTLIST SensorConfig name     CDATA #IMPLIED
                       function CDATA #IMPLIED
                       sensor   CDATA #REQUIRED
                       unit     CDATA #IMPLIED>
```

An example document that describes an actual experiment setting could look like the following one:

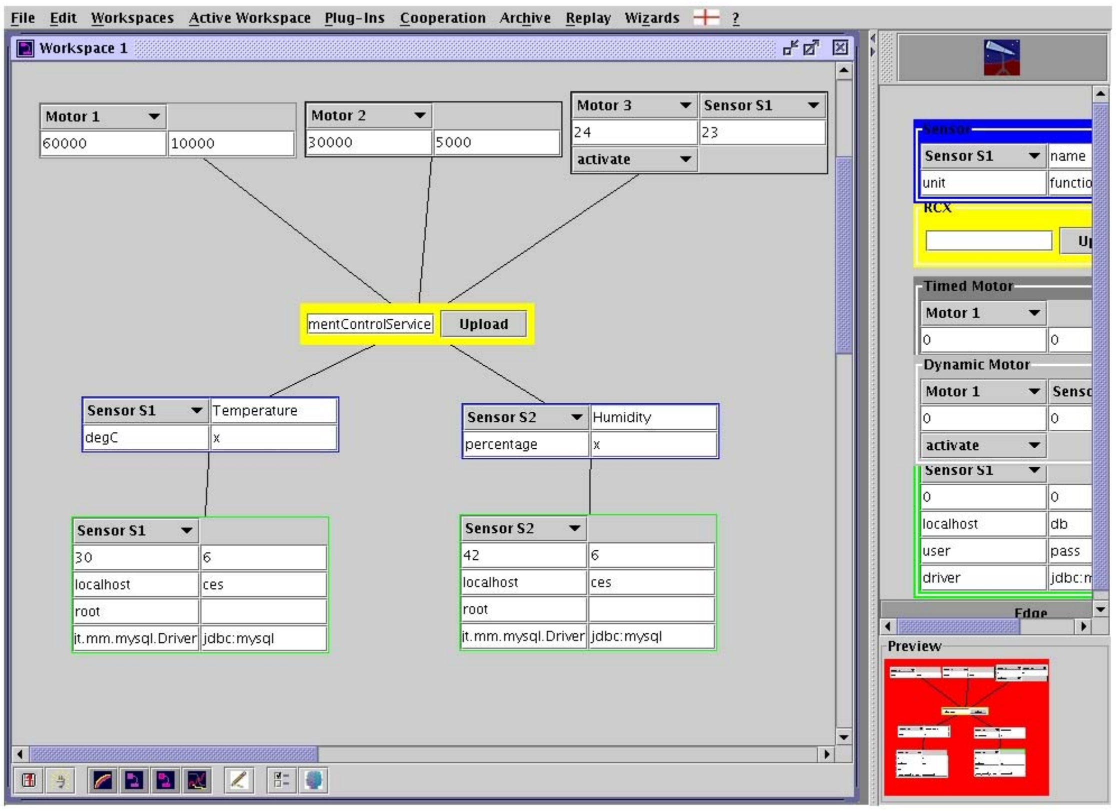
```
<?xml version="1.0"  encoding="UTF-8"?>

<!DOCTYPE Experiment SYSTEM "http://www.coldex.info/control.dtd">

<Experiment>
  <Motor motor="1">
    <Timed each="60000" for="10000" />
  </Motor>
  <Motor motor="2">
    <Timed each="30000" for="5000" />
  </Motor>
  <Motor motor="3">
    <Dynamic sensor="1" startAction="24" stopAction="23"
                                         action="activate" />
  </Motor>
  <DataLogging sensor="1" each="30" numberOfValues="6">
    <DatabaseConnection protocol="jdbc:mysql" database="ces"
                        driver="org.gjt.mm.mysql.Driver" pass=""
host="localhost" user="root" />
  </DataLogging>
  <DataLogging sensor="2" each="42" numberOfValues="6">
    <DatabaseConnection protocol="jdbc:mysql" database="ces"
                        driver="org.gjt.mm.mysql.Driver" pass=""
host="localhost" user="root" />
  </DataLogging>
  <Configurations>
```

```
    <MotorConfig name="Waterpump" motor="1" speed="9" />
    <MotorConfig name="Rotator" motor="2" speed="9" />
    <MotorConfig name="Fans" motor="3" speed="9" />
    <SensorConfig name="Temperature" function="x" sensor="1"
                  unit="degC" />
    <SensorConfig name="Humidity" function="x" sensor="2"
                  unit="percentage" />
  </Configurations>
</Experiment>
```

These XML descriptions of an experiment setting are to be transferred as a string to the WebService, which actually passes it directly to the experiment control server where it is parsed and translated to the needed Java classes.

## 4. The Cool Modes reference frame

As already stated, a Cool Modes reference frame was created to allow for the visual construction of an experiment description. Basically, the reference frame has a node for each of the components needed to define an experiment plan that is valid according to the DTD. Figure 2 shows a visual representation of the above shown XML file.


File Edit Workspaces Active Workspace Plug-Ins Cooperation Archive Replay Wizards ?
Workspace 1
Upload
Sensor
RCX
Up
Timed Motor
Dynamic Motor
Edge
Preview

Motor 1 | 60000 | 10000
Motor 2 | 30000 | 5000
Motor 3 | Sensor S1 | 24 | 23 | activate
mentControlService
Sensor S1 | Temperature | degC | x
Sensor S2 | Humidity | percentage | x
Sensor S1 | 30 | 6 | localhost | ces | root | it.mm.mysql.Driver | jdbc:mysql
Sensor S2 | 42 | 6 | localhost | ces | root | it.mm.mysql.Driver | jdbc:mysql
Sensor S1 | name | unit | functio
Motor 1 | 0 | 0
Motor 1 | Senso | 0 | 0 | activate
Sensor S1 | 0 | 0 | localhost | db | user | pass | driver | jdbc:m

**Figure 2: Cool Modes with a visual experiment description**

## 5. Future Efforts

To provide a starting point for changing the experiment plan the system could:/

- Retrieve current experiment from the control demon and reconstruct the coolmodes workspace.

- Retrieve previously implemented experiment plans from a repository.

CoolModes interface improvement:

- Collaborative (Matchmaker)?
- Webcam pictures
- Update visual components
    - Eg. Use an image for the RCX

Alerting system

- Informational/warning/critical conditions thresholds
- email daily digest (link to daily itool)
- html daily and weekly summary sites (embedding itool and pictures)

TI – data logging

- direct access?
- Datafile parsing?

Leonardo integration

*Demo scenario (possibly for the COLDEX review)*

- Inquiry tool update for live data plotting
- Physical spray bottle to actuate humidity changes
    - And/Or CO2 injection
- Sensor calibration
- DataLogging stability

Storage of experiments into database/LOR to provide a sequence of the experiment plans over time and for retrieval via coolmodes and/or inquiry tool.

Configuration of the DataLogging in the DTD should include a way to flexibly configure the SiteID of the site – currently this value is hardcoded to 0. SensorID should similarly be configurable in the case of a single database system for multiple sites. This functionality is less important for this initial example with a single site and known sensors.

Testing and debugging and stability reinforcement

DataLogging calibration and testing the averaging functionality