iCasa Tutorial

https://self-star.imag.fr

Philippe Lalanda, Julie McCann & Ada Diaconescu

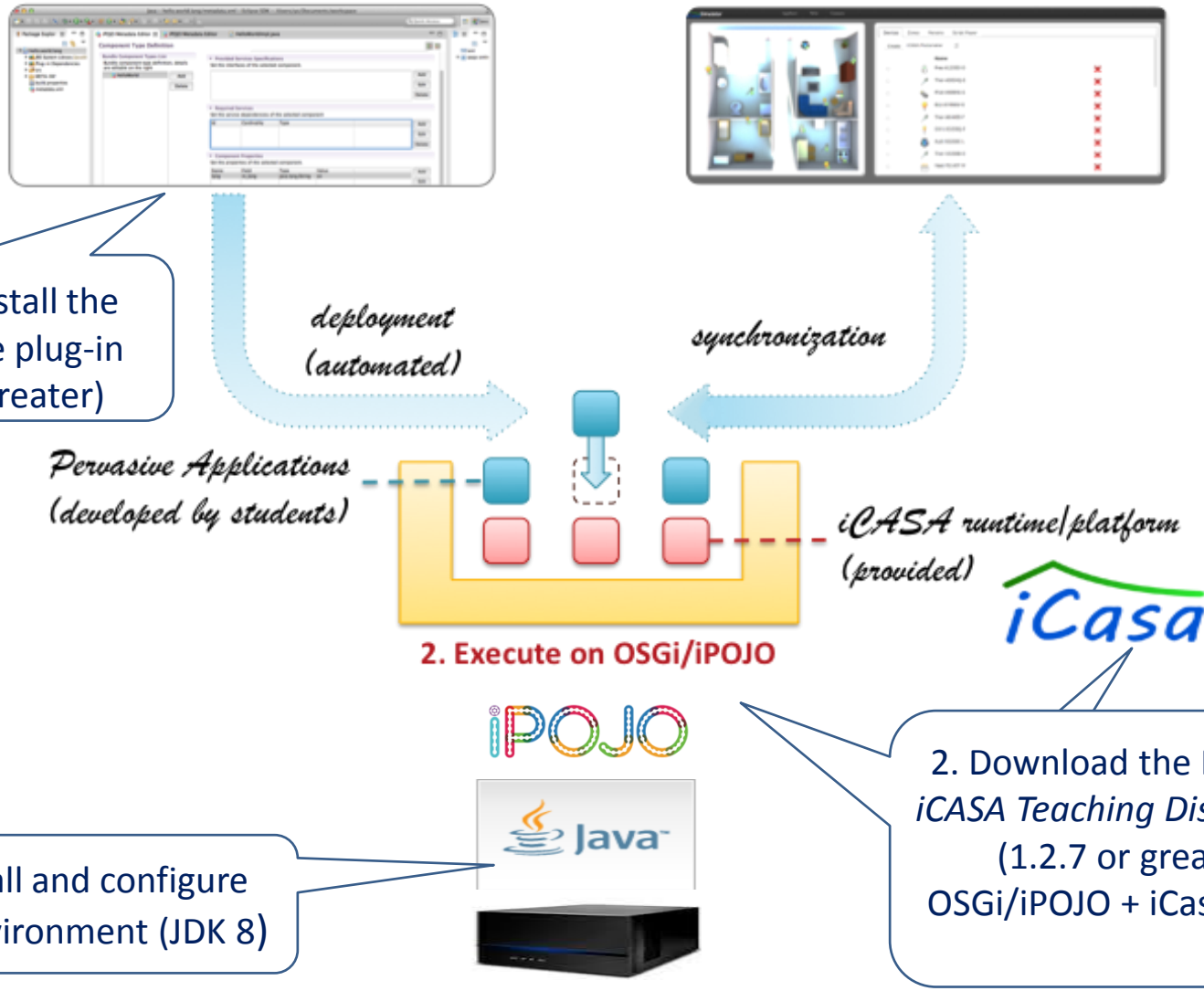
UNIVERSITÉ Grenoble Alpes

Imperial College London

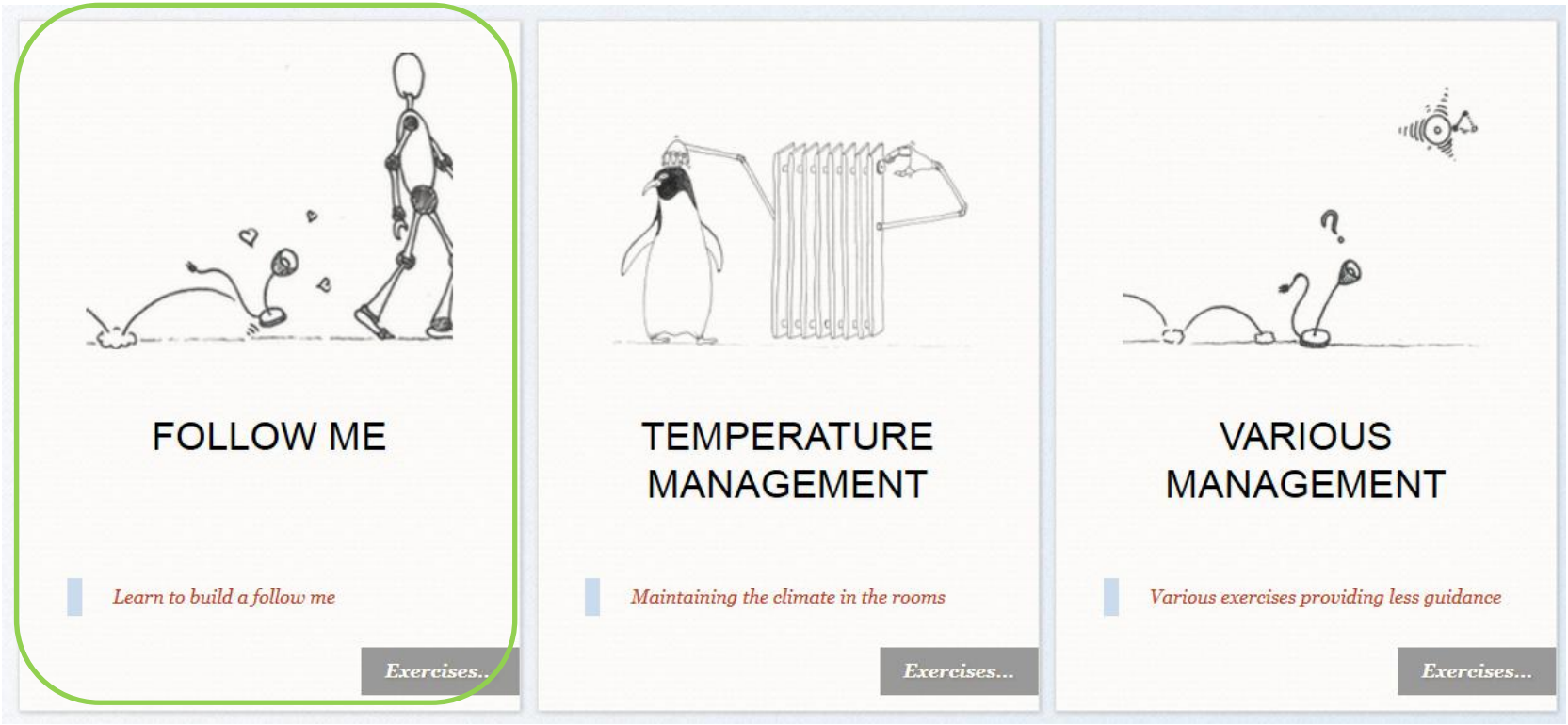
INSTITUT Mines-Télécom

TELECOM ParisTech

Setting-up the environment



Main exercises



Today's Tutorial: *Light* Management

Overview

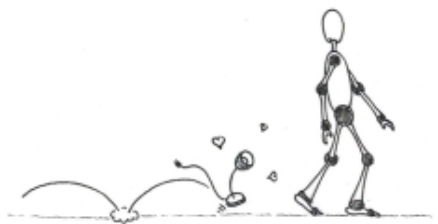
- Part 1

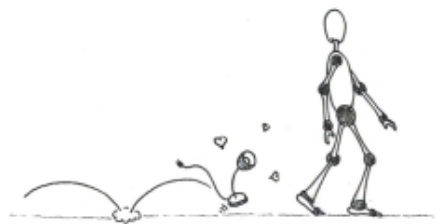
  Step-by-step introduction to iCASA

  Web console, IDE & Runtime environment

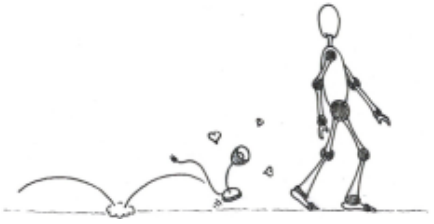
- Part 2

  Illustration of exercises

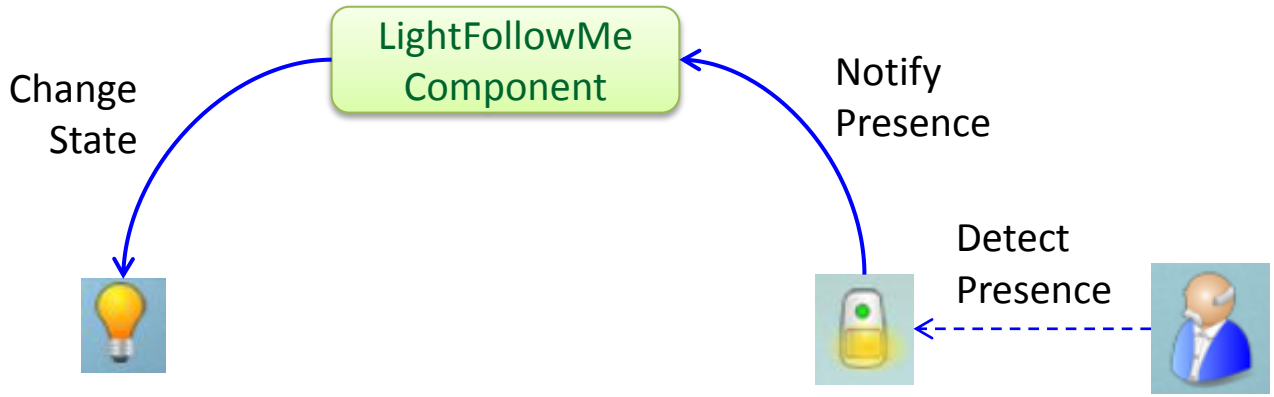




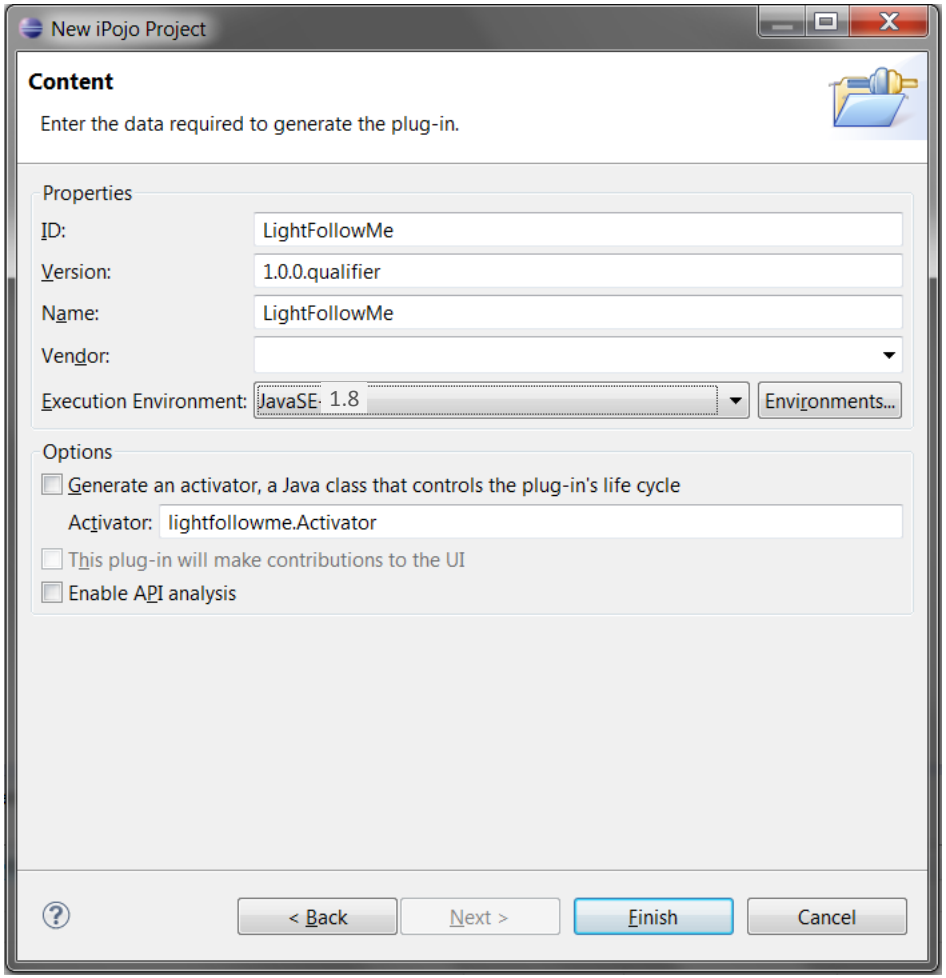
Part 1

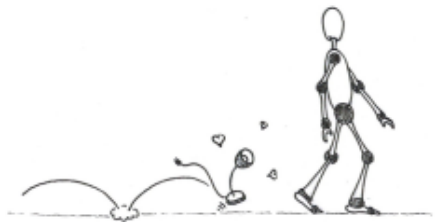


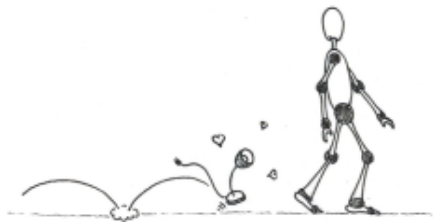
- Create an *iPOJO project*
- Create a simple *LightFollowMe component*
- Add basic *Monitoring* and *Adaptation* logic to enable the LightFollowMe component to manage iCASA devices



Create an iPOJO Project: LightFollowMe







Use the iPOJO Metadata & Java Editors to write the metadata.xml & Java class

- Create a LightFollowMe *component*
- Create service *dependencies* to BinaryLights and PresenceSensors
- Specify *lifecycle* methods (start & stop)
- Generate *Java class* LightFollowMeImpl.java
- Implement un/bind and lifecycle start/stop methods (Java editor)
- Create an *instance* of the LightFollowMe component

Start iCASA, Deploy bundle & Check

- Start iCASA Runtime
  - Go to the tutorial distribution's <Home> directory
  - Run startGateway.sh/bat

- Deploy and test the LightFollowMe bundle
  - LightFollowMe project → iCASA → Bundle Deployment

  ⇒ <Home>/applications/LightFollowMe.jar

- Check bundle installation:
  - Command line → lb

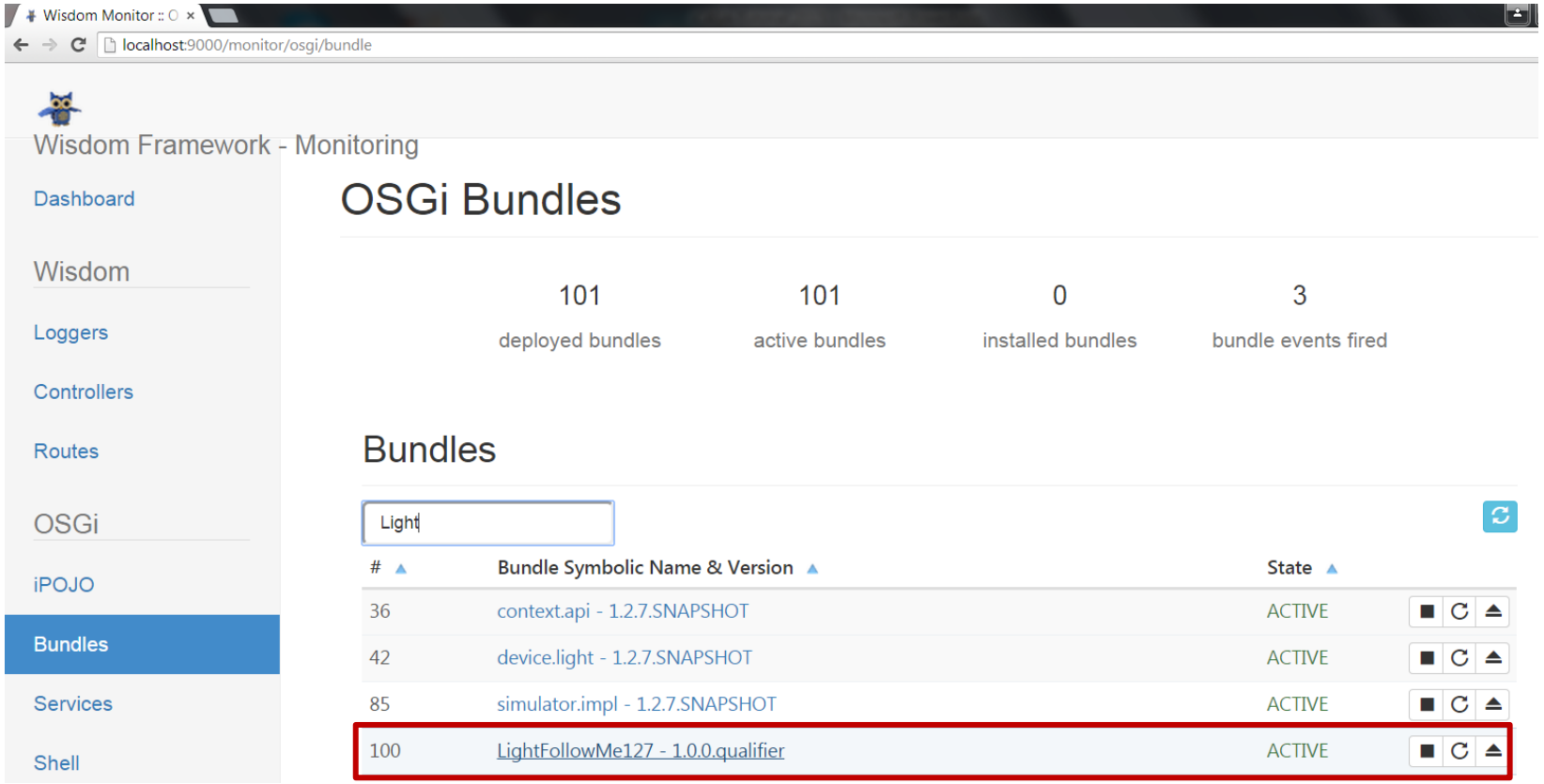
```
g! lb
START LEVEL 1
   ID|State      |Level|Name
    0|Active     |    0|System Bundle (4.0.3)
   70|Active     |    1|iCasa :: device.light (1.2.4)
   71|Active     |    1|iCasa :: device.motion (1.2.4)
   72|Active     |    1|iCasa :: context.impl (1.2.4)
   73|Active     |    1|iCasa :: common (1.2.4)
   74|Active     |    1|LightFollowMe (1.0.0.qualifier)
g!
```

iCASA web console

- http://localhost:9000/simulator

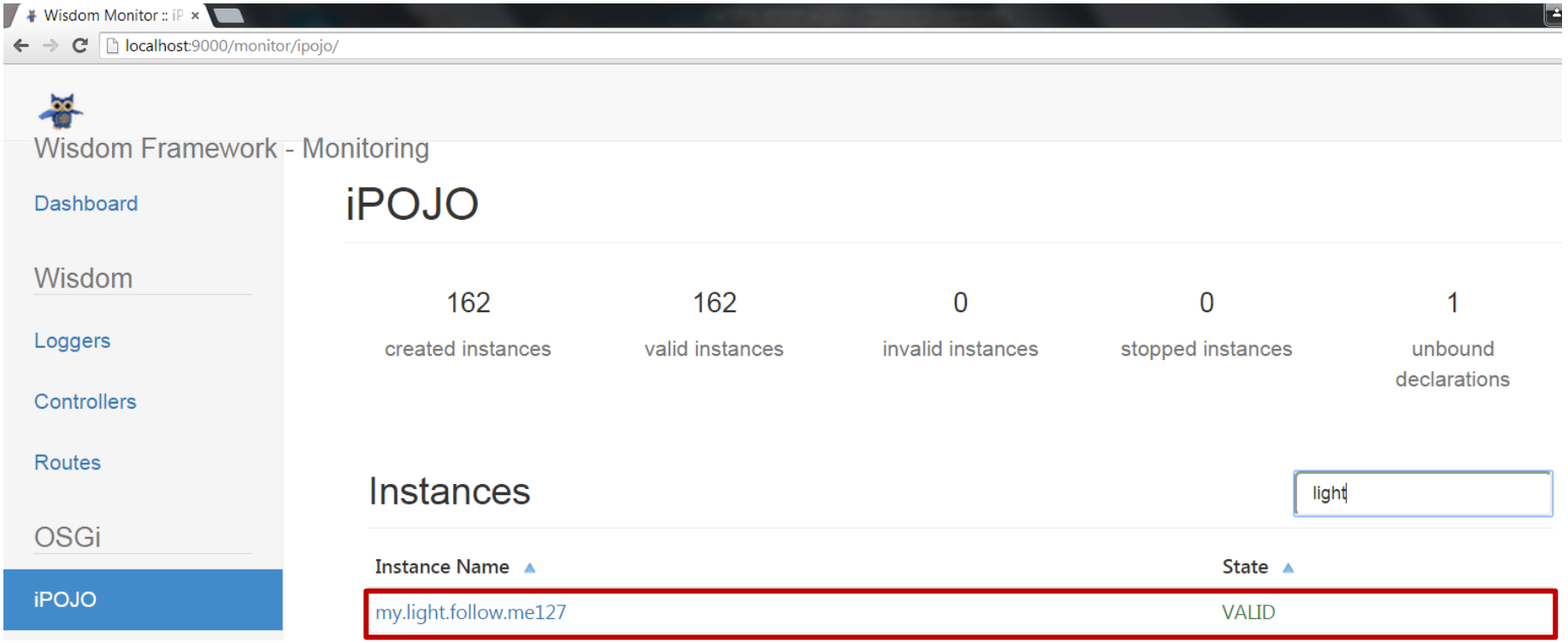
iPOJO web console → bundles

- http://localhost:9000/monitor (Note: avoid Firefox)
  - /osgi/bundles
  - usr&pw: 'admin' ☺



iPOJO web console → iPOJO

- http://localhost:9000/monitor
  - /iPOJO



Check lifecycle and un/bind methods

- Command line: start & stop LightFollowMe bundle
  - `start <bundle_id>` => you should see your *start message*...
  - `stop <bundle_id>` => you should see your *stop message*...

```
   74|Active     |    1|LightFollowMe (1.0.0.qualifier)
g! stop 74
LightFollowMe Component is stopping...
g! start 74
g! LightFollowMe Component is starting...
```

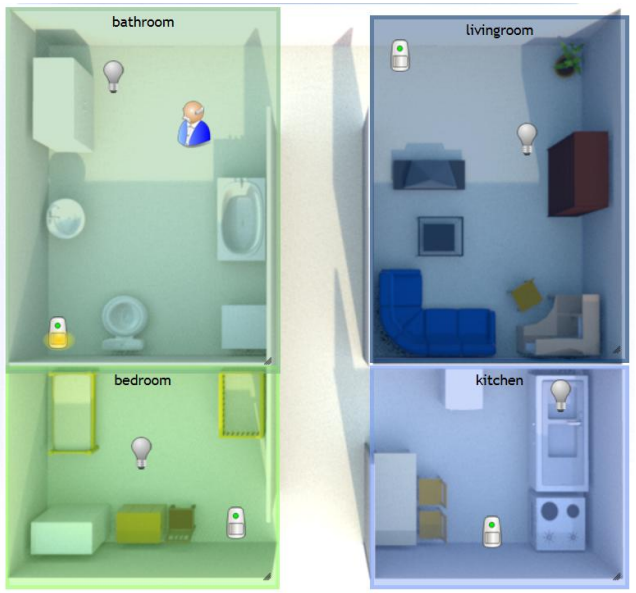
- iCASA web console > Device tab: add & remove devices
  - add a BinaryLight device => your *bind message*...
  - add a PresenceSensor device => your *bind message* ...
  - remove the BinaryLight device => your *unbind message*...
  - remove the PresenceSensor device => your *unbind message*...

```
bind binary light BinaryLight-7015a6b604
bind presence sensor PresenceSensor-745336aaf9
unbind binary light BinaryLight-7015a6b604
unbind presence sensor PresenceSensor-745336aaf9
```

Use scripts to ‘populate’ your environment automatically

- Scripts (`.bhv` files)
  → `<iCASA-Home>/scripts` directory

- iCASA web console – Script Player tab
  → Select and start a script:
  `single_bl_light_environment.bhv`




```
<create-zone id="kitchen"  leftX="410" topY="370" X-Length="245" Y-Length="210" />

<create-device id="Pres-A1255D-D" type="iCasa.PresenceSensor" />
<create-device id="BiLi-A7496W-S" type="iCasa.BinaryLight" />
<move-device-zone deviceId="Pres-A1255D-D" zoneId="kitchen" />
<move-device-zone deviceId="BiLi-A7496W-S" zoneId="kitchen" />
 ...
```

iCASA *Monitoring* via notifications and listeners

- Implement `DeviceListener` interface
  - To get notified of device changes – e.g. presence detection
  - Each device has specific properties - e.g.:
    - `PresenceSensor.PRESENCE_SENSOR_SENSED_PRESENCE`
    - `PresenceSensor.LOCATION_PROPERTY_NAME`
- Register listeners with devices
  - e.g. `presenceSensor.addListener(<listener-instance>);`
- Deploy & test by moving a person across rooms

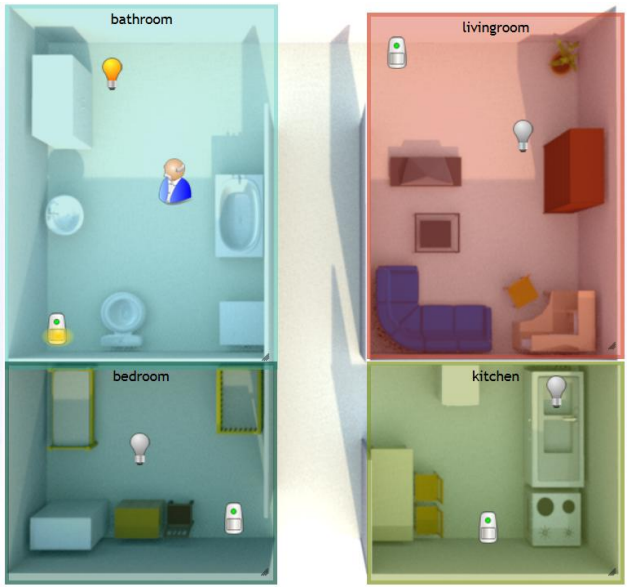
```
g! device property 'presenceSensor.sensedPresence' was modified
        > from 'true' to 'false'
        > for device of type: fr.liglab.adele.icasa.device.presence.impl.SimulatedPresenceSensorImpl
        > with device id: Pres-B1255D-D
device property 'presenceSensor.sensedPresence' was modified
        > from 'false' to 'true'
        > for device of type: fr.liglab.adele.icasa.device.presence.impl.SimulatedPresenceSensorImpl
        > with device id: Pres-D1255D-D
```

iCASA *Adaptation* via device changes

- React to notifications by adapting devices
  - Modify device states - e.g. light on/off status
  - Each device has a specific control interface – e.g.
    - `binaryLight.setPowerStatus(true);`

- E.g. switch lights on or off when presence or absence is detected, respectively
  - Get all lights at location
  - Manage light status (on/off)

- Deploy & Test

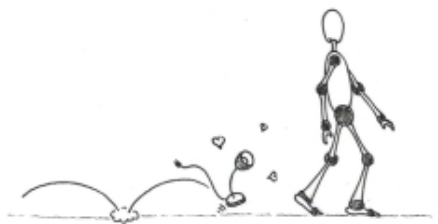


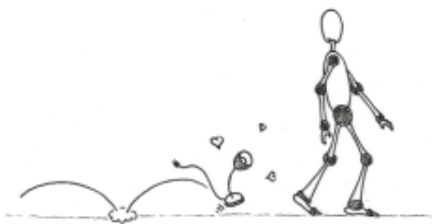
Part 2 - exercises

1. Manage *location changes* of Binary Lights

2. Managing *multiple* Binary Lights per room

3. Providing an *Administration service*





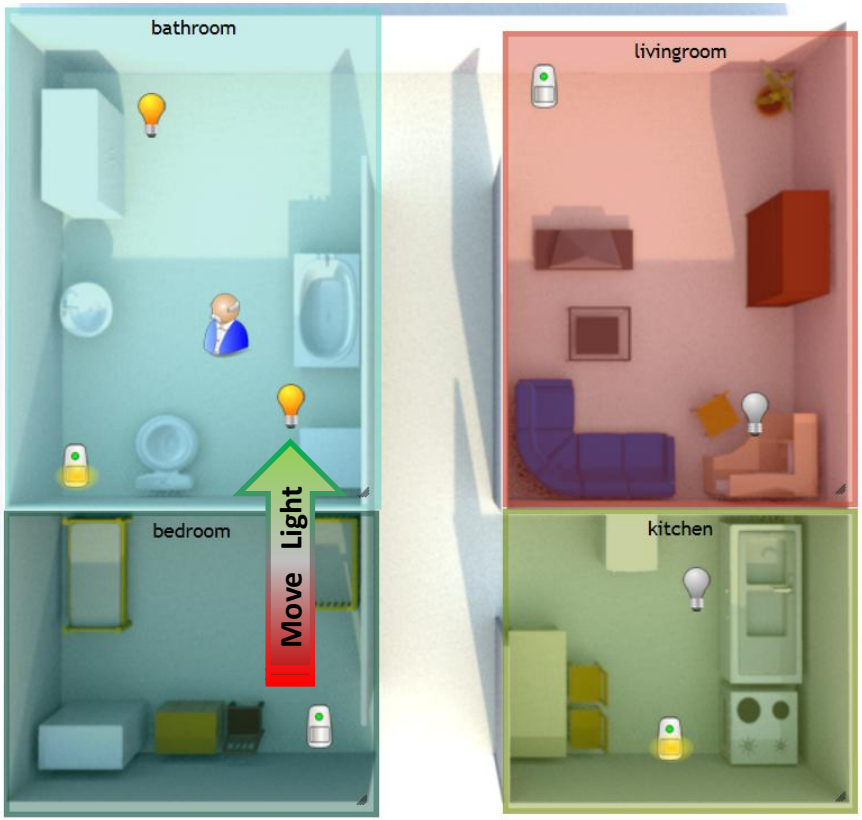
Exo1: managing light movement

- Handle events issued by BinaryLights when moved
  - Event name: `LOCATION_PROPERTY_NAME`

- If presence sensed at the new location then switch-on light

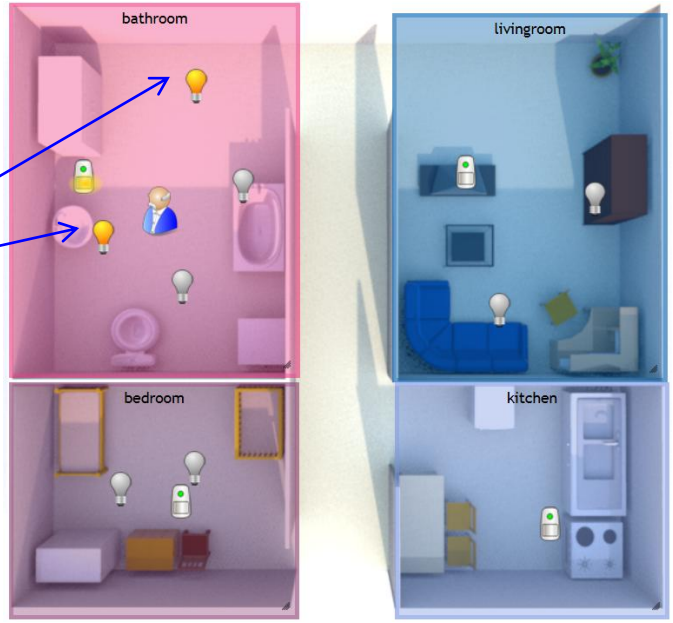
- Else switch-off light

- Deploy & test



Exo2: managing multiple lights

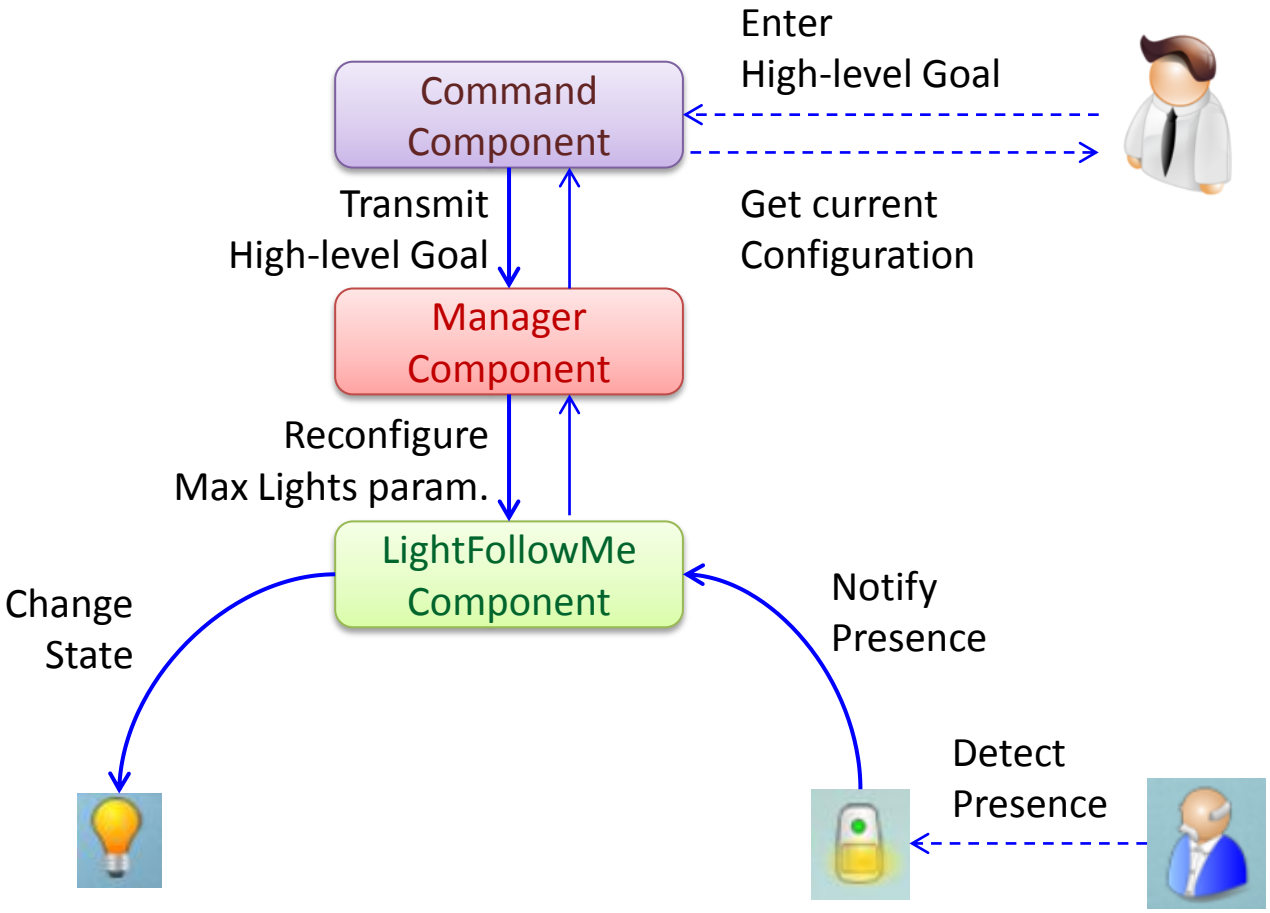
- Always try to switch-on
  a targeted number of lights
  when detecting a presence
  - E.g.
    `maxLightsOnPerRoom = 2`



- When a light leaves a room
  - also manage the remaining lights
    in that room

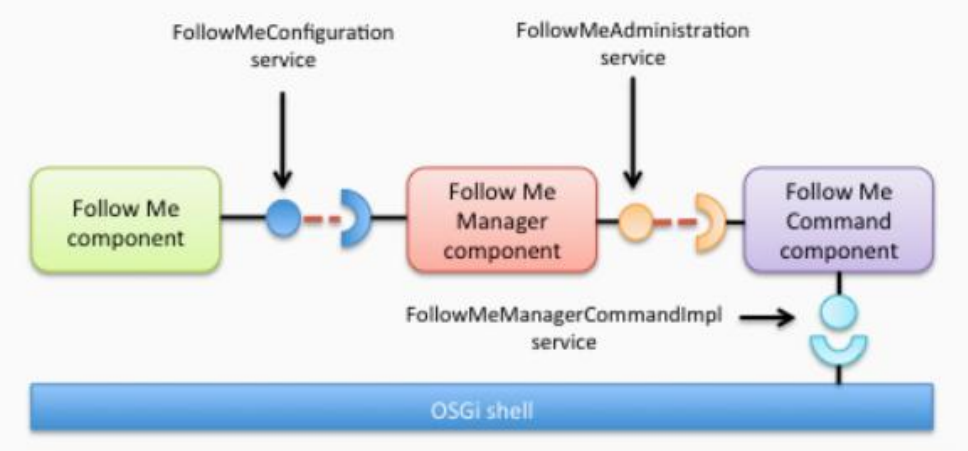
- Use another script to create more lights
  - E.g.
    `multiple_lights_environment.bhv`

Exo3: adding an administration service



Exo3: administration service (2)

- Instrument the `LightFollowMe` component to allow *monitoring* and *reconfiguration*
  - Provide the `FollowMeConfiguration` interface/service
- Add a `Manager` service
  - Follows high-level goals: High, Medium or Low light intensity
  - Translates high-level goals into lower-level configurations
- a `Command` service
  - Receives the high-levels goals from the System Administrator, via the command console




```
75|Active      |    1|LightFollowMe (1.0.0.qualifier)
76|Active      |    1|Manager (1.0.0.qualifier)
77|Active      |    1|Command (1.0.0.qualifier)
```

Exo3: administration service (3)

- Set & get light intensity goals (via the command line)

```
g! setIlluminancePreference HIGH
INFO: LightFollowMe: re-set the maximumNumberOfLights attribute to 3
```

```
g! getIlluminancePreference
The illuminance goal is FULL
```


```
g! setIlluminancePreference SOFT
INFO: LightFollowMe: re-set the maximumNumberOfLights attribute to 1
```

Thank you!

Questions?

Bookmarks

- Autonomic Computing book
  - https://www.springer.com/computer/swe/book/978-1-4471-5006-0
- Tutorial website
  - https://self-star.imag.fr
- iCASA website
  - http://adeleresearchgroup.github.io/iCasa
- iPOJO website
  - http://ipojo.org