The UniversAAL Platform

Alexander Kocian

Department of Computer Science
University of Pisa
Largo B. Pontecorvo 3
56127 Pisa

2014/2015

Table of Contents

1. Introduction
2. From Challenges to Solutions
3. OSGi component based platform
4. Middleware
5. Experimentation with the UniversAAL Platform

What is UniversAAL ?[1]

In fact, UNIVERsal open platform and reference Specification for Ambient Assisted Living is a piece of software.

Definition

UniversAAL is an open-source software platform for AT where various, **heterogeneous** technical devices may be connected to a single, unified network.

Alert

The MS Windows and Apple MacOS platforms are only able to handle **homogeneous** technical devices.

Devices

The technical devices are either sensors or actuators or both.

- Sensors provide the system with information about the current state of the environment (so-called "contextual information"). Examples: pressure sensor, motion sensor, brightness sensor, camera, clock,...
- Actuators can be used by the system to influence the current state of the environment. Examples: heater, TV, electric window,...

Support Platform

The universAAL platform is called a Platform, because it is more than just a software layer that lies **between operating system and the applications in a distributed computer network** (aka "Middleware)

- **Runtime Support** (Implementation of the Execution Environment)
- **Development Support** (a suite of SW tools for supporting the SW developer)
- **Community Support** (a suite of SW facilities and technical infrastructure to assist end users, service providers and developers in community-building)

A Layer Representation of the Platform

- The platform can logically be divided into various layers: Middleware, Managers, Applications.

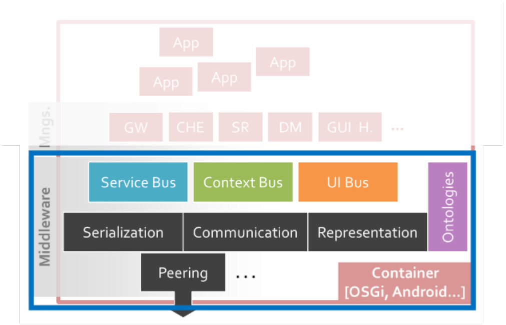


Figure: Layered Model[2]

The Middleware Layer

- It needs to be available on every active node.
- Its task is to hide the distribution and hetereogenity of the nodes.
- Each communication bus (Context-Bus, Service-Bus, User-Interaction-Bus) handles a specific type of message.

The Middleware (cont'd)

- The Context-Bus is responsible for sharing context information, i.e. sharing knowledge that is used to dynamically adapt services from application to the user and vice versa[3].

Examples of context

identity, location (geographical data), status (temperature, ambient illumination, noise level) and time[4].

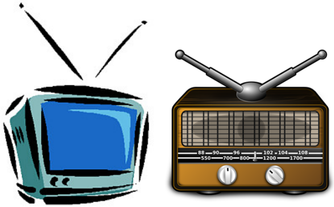
- The Service-Bus is responsible for sharing access to the service, i.e. sharing functionality.
- The User-Interaction-Bus is responsible for sharing information to active user interaction.

The Application Platform

The challenge - running applications on multiple hetereo-geneous devices.

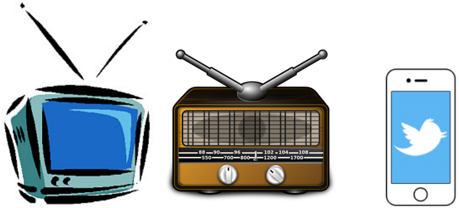
The Application Platform

The challenge - running applications on multiple hetereo-geneous devices.



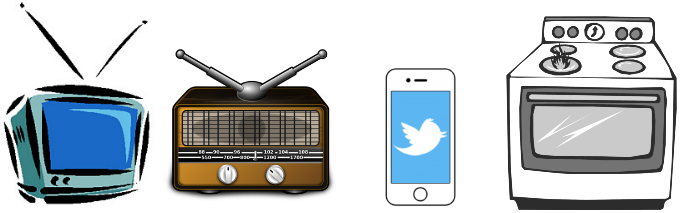
The Application Platform

The challenge - running applications on multiple hetereogeneous devices.



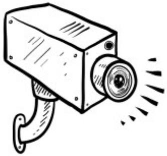
The Application Platform

The challenge - running applications on multiple hetereo-geneous devices.



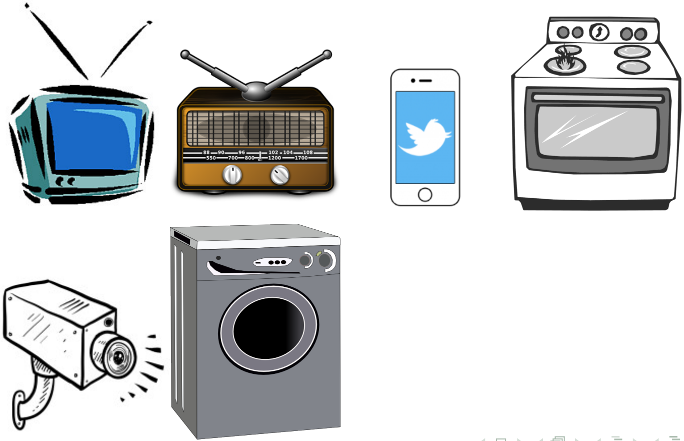
The Application Platform

The challenge - running applications on multiple hetereo-geneous devices.



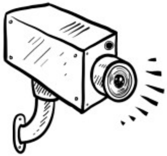
The Application Platform

The challenge - running applications on multiple hetereo-geneous devices.



The Application Platform

The challenge - running applications on multiple hetereo-geneous devices.

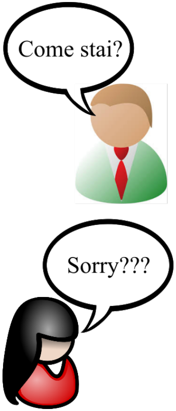


The Application Platform

The challenge - running applications on multiple hetereo-geneous devices.

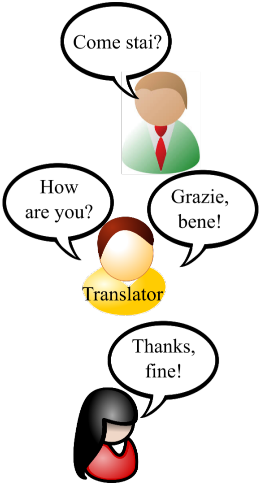
Heterogeneity of the devices

- Independent development and production of consumer items.
- Ability to exchange data depends on
  - Networking protocol (switching and routing)
  - Access protocol (synchronization,FEC)
  - Data representation (compression, encryption)
- Several application domains
- Several standards per application domain
- Several application profiles per standard
- What to do if all are relevant?



Middleware solutions

- For "AAL" components, a main protocol for networking & communication, optimally based on a single solution for data representation
- Integration of legacy components through adapters
  - Networking layer: protocol-specific gateways
  - Link and Presentation layers: component-specific wrappers



Challenges

Devices can come and go

Challenges

Devices can come and go

- Mobile devices - smart phones, body sensors, portable audio players

Challenges

Devices can come and go

- Mobile devices - smart phones, body sensors, portable audio players
- can be switched on and off

Challenges

Devices can come and go

- Mobile devices - smart phones, body sensors, portable audio players
- can be switched on and off
- can fail and be restarted

Challenges

Devices can come and go

- Mobile devices - smart phones, body sensors, portable audio players
- can be switched on and off
- can fail and be restarted

Applications can come and go

Challenges

Devices can come and go

- Mobile devices - smart phones, body sensors, portable audio players
- can be switched on and off
- can fail and be restarted

Applications can come and go

- can be installed, updated, uninstalled

Challenges

Devices can come and go

- Mobile devices - smart phones, body sensors, portable audio players
- can be switched on and off
- can fail and be restarted

Applications can come and go

- can be installed, updated, uninstalled
- can fail and be restarted

Challenges

Devices can come and go

- Mobile devices - smart phones, body sensors, portable audio players
- can be switched on and off
- can fail and be restarted

Applications can come and go

- can be installed, updated, uninstalled
- can fail and be restarted

It is **not feasible to restart** the platform for any change in a device/an application.

Challenges

Devices can come and go

- Mobile devices - smart phones, body sensors, portable audio players
- can be switched on and off
- can fail and be restarted

Applications can come and go

- can be installed, updated, uninstalled
- can fail and be restarted

It is **not feasible to restart** the platform for any change in a device/an application. The platform and the application should auto-**adapt** to any change.

The Solution: Open Service Gateway initiative (OSGi)[5]

The Solution: Open Service Gateway initiative (OSGi)[5]

- is a specification.

The Solution: Open Service Gateway initiative (OSGi)[5]

- is a specification.
- The core of the spec defines a **component and service** model for Java ®.

The Solution: Open Service Gateway initiative (OSGi)[5]

- is a specification.
- The core of the spec defines a **component and service** model for Java ®.
- Components and services (i.e. Java interfaces) can be **dynamically** installed, started, stopped, updated and uninstalled **without restarting the container**.

The Solution: Open Service Gateway initiative (OSGi)[5]

- is a specification.
- The core of the spec defines a **component and service** model for Java ®.
- Components and services (i.e. Java interfaces) can be **dynamically** installed, started, stopped, updated and uninstalled **without restarting the container**.
- OSGi has several implementations, such as Equinox, Knopflerfish OSGi or **Apache Felix**.

OSGi Bundles

- Services are packaged into bundles.

OSGi Bundles

- Services are packaged into bundles.
- Bundles are a cohesive, self-contained units of functionality.

OSGi Bundles

- Services are packaged into bundles.
- Bundles are a cohesive, self-contained units of functionality.
- Technically, OSGi bundles are `.jar` files with additional meta information (images, libraries,...), stored in `MANIFEST.MF` file.

OSGi Bundles

- Services are packaged into bundles.
- Bundles are a cohesive, self-contained units of functionality.
- Technically, OSGi bundles are `.jar` files with additional meta information (images, libraries,...), stored in `MANIFEST.MF` file.
- Dependencies to other modules and services are explicitly defined via `MANIFEST.MF`.

OSGi Bundles

- Services are packaged into bundles.
- Bundles are a cohesive, self-contained units of functionality.
- Technically, OSGi bundles are `.jar` files with additional meta information (images, libraries,...), stored in `MANIFEST.MF` file.
- Dependencies to other modules and services are explicitly defined via `MANIFEST.MF`.
- Any non-OSGi runtime ignores the OSGi metadata.

OSGi Bundles

- Services are packaged into bundles.
- Bundles are a cohesive, self-contained units of functionality.
- Technically, OSGi bundles are `.jar` files with additional meta information (images, libraries,...), stored in `MANIFEST.MF` file.
- Dependencies to other modules and services are explicitly defined via `MANIFEST.MF`.
- Any non-OSGi runtime ignores the OSGi metadata.
- OSGi bundles have a life-cycle.

Bundle Lifecycle

- With `install <.jar>` in the OSGi runtime, the bundles are presisted in a local cache. A bundle ID is returned.
- With `resolve`, bundle dependencies are resolved.
- More bundles can be installed and resolved.

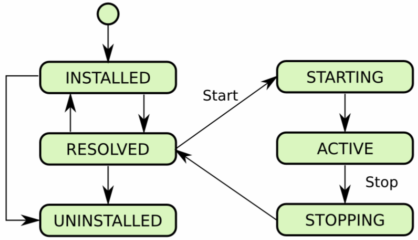


Figure: State Diagram of the Bundle life cycle

Bundle Lifecycle (cont'd)

- Next, `start <bundle id>`.
- The bundle is now run-nig i.e., in active state.
- With `stop <bundle id>`, the bundle is still in the local bundle cache.
- `uninstall <bundle id>`, to remove the bundle from the cache.

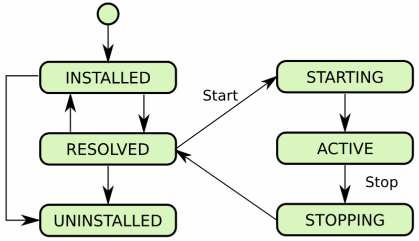


Figure: State Diagram of the Bundle life cycle

Complexity of Software

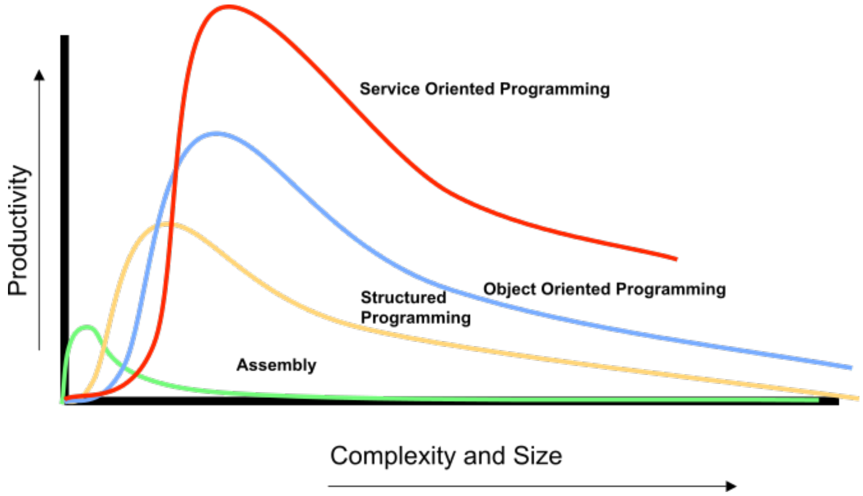


Figure: Complexity of SW[6]

OSGi - a service oriented architecture

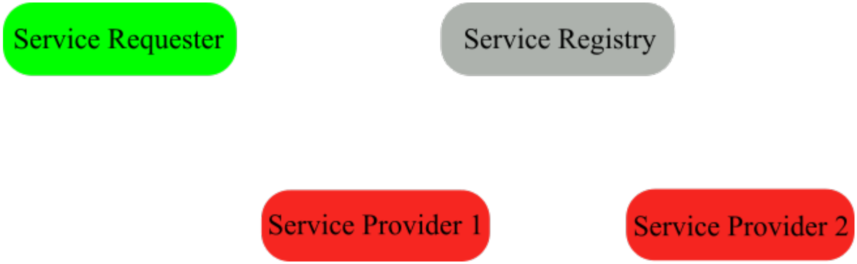


Figure: Pattern for service-oriented component model[7]

- An OSGi Service is defined by a standard Java® class or interface.

OSGi - a service oriented architecture

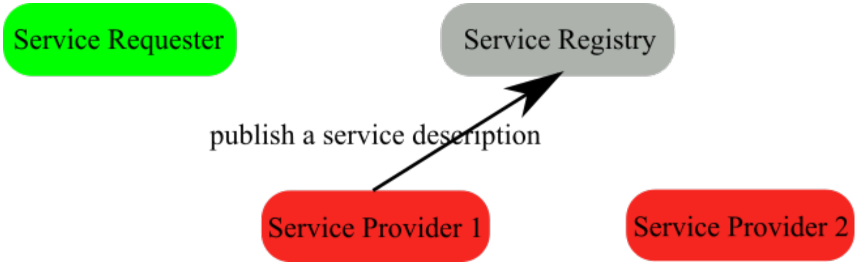


Figure: Pattern for service-oriented component model[7]

- A bundle can register and use OSGi services.

OSGi - a service oriented architecture

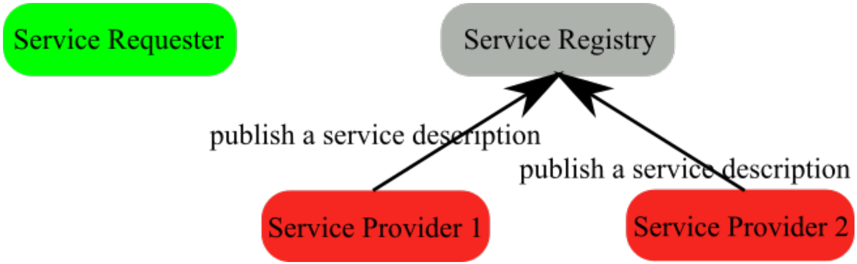


Figure: Pattern for service-oriented component model[7]

- Another bundle can register and use OSGi services.

OSGi - a service oriented architecture

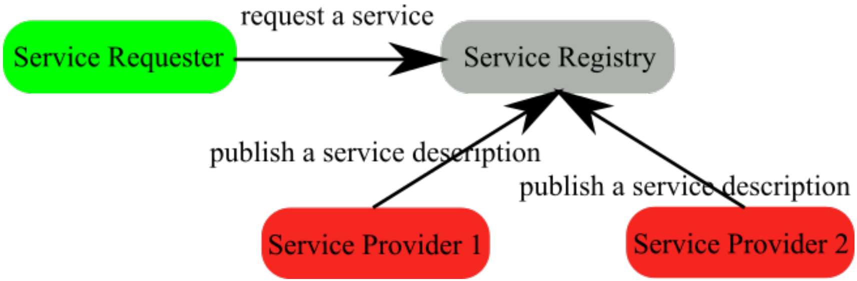


Figure: Pattern for service-oriented component model[7]

- A service is requested.

OSGi - a service oriented architecture

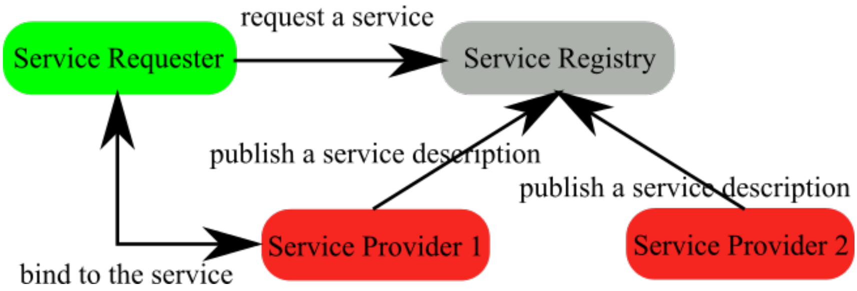


Figure: Pattern for service-oriented component model[7]

- If several services are valid for the same API, then OSGi chooses that with lowest service ID.

OSGi - a service oriented architecture

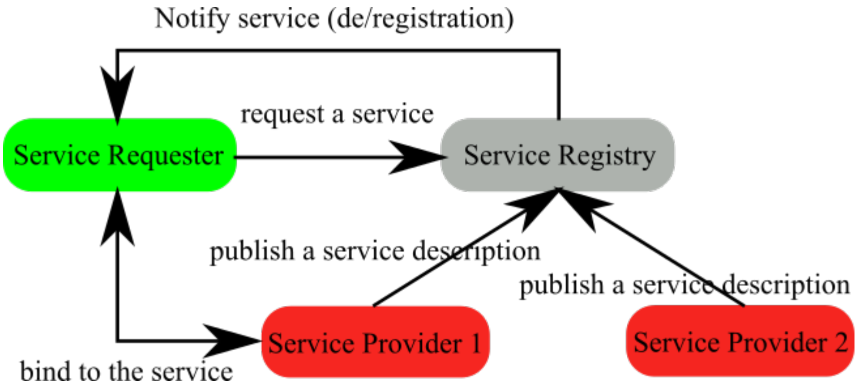
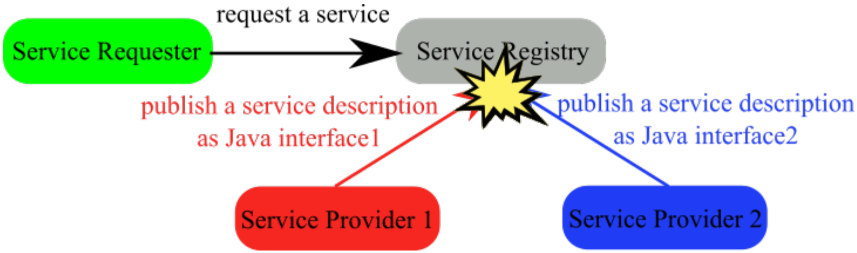


Figure: Pattern for service-oriented component model[7]

- Service providers can be switched on the fly.

Interoperability Problem

- The Service Requester and all Service Providers have to agree a priori on **exactly** the same service interface.
- Mismatch otherwise.



Semantic Services

Solution

Instead of directly connecting service provider with service interface, we apply reasoning using **ontology.**

Ontology in UniversAAL

Definition

- Ontology (from Greek: $o\nu\tau o\lambda o\gamma\iota\alpha$) is the philosophical study of the nature of being.
- In computer science, an ontology is an "explicit specification of a conceptionalization" [8]. Simply, a model of the real world so that information in the model can be processed by computers.

Purpose

- Distribution of knowledge (*Context Bus* in uAAL)
- Sharing of functionalities (*Service Bus* in uAAL)

Distribution of Knowledge

Two apps that share knowledge interpret info by ontology in **exact** the same way.

Construction

- Ontologies are made up of classes, properties, and data types.
- Every ontology has a uniform resource identifier URI.

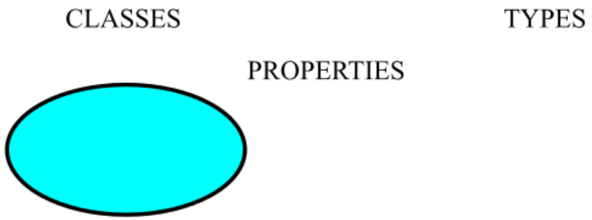
A Taste of Resource Description Framework (RDF)

CLASSES

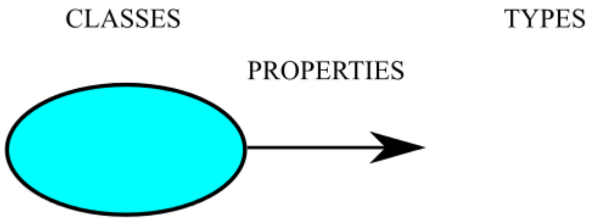
TYPES

PROPERTIES

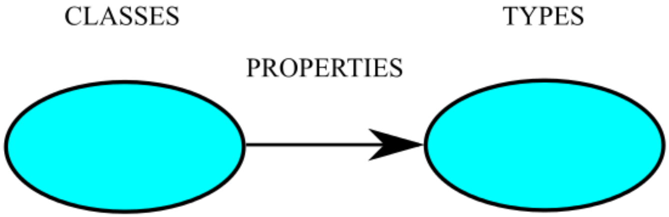
A Taste of Resource Description Framework (RDF)



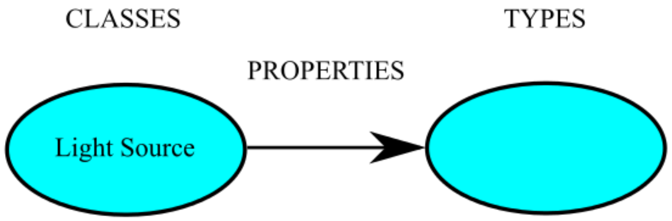
A Taste of Resource Description Framework (RDF)



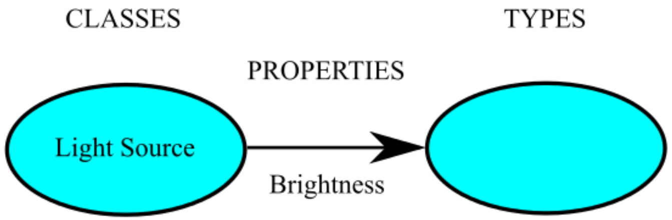
A Taste of Resource Description Framework (RDF)



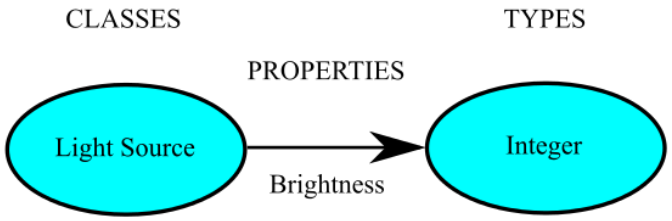
A Taste of Resource Description Framework (RDF)



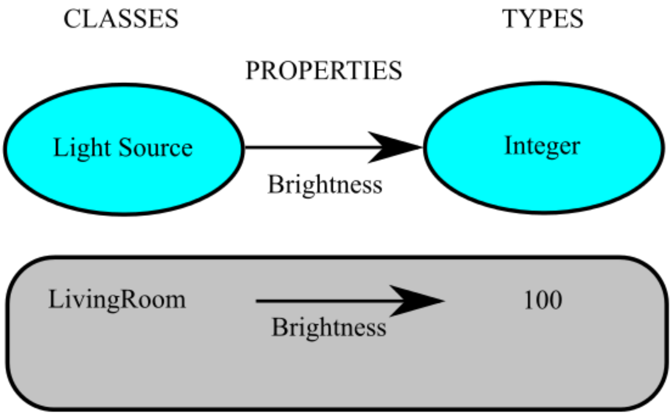
A Taste of Resource Description Framework (RDF)



A Taste of Resource Description Framework (RDF)



A Taste of Resource Description Framework (RDF)

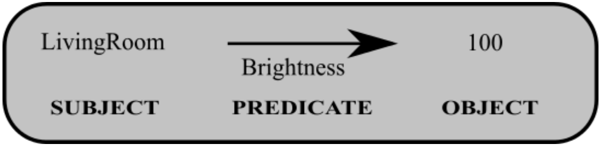


RDF Statement

Definition

- An RDF statement is a **triple** (subject, predicate, object)
- All subjects of RDF staements are resources with **Unique Resource Identifier** (URI)

Example



http://ontology.universaal.org/Lighting.owl#LightSource

Implementation in UniversAAL

```
public class LightSource extends PhysicalThing
{
public static final String MY_URI =
"http://ontology.persona.ima.igd.fhg.de/Lighting.owl#LightSource";
public static final String PROP_AMBIENT_COVERAGE =
"http://ontology.persona.ima.igd.fhg.de/Lighting.owl#ambientCoverage"
public static final String PROP_HAS_TYPE =
"http://ontology.persona.ima.igd.fhg.de/Lighting.owl#hasType";
public static final String PROP_SOURCE_BRIGHTNESS =
"http://ontology.persona.ima.igd.fhg.de/Lighting.owl#srcBrightness";
}
```

Non-OSGi devices

The Problem

- JVM does not exist on every device;
- OSGi-like module framework for C does not emulate Java®features (bytecode, classloading,...);
- ergo, OSGi cannot be installed on every device.

The Solution

Non-OSGi devices

The Problem

- JVM does not exist on every device;
- OSGi-like module framework for C does not emulate Java®features (bytecode, classloading,...);
- ergo, OSGi cannot be installed on every device.

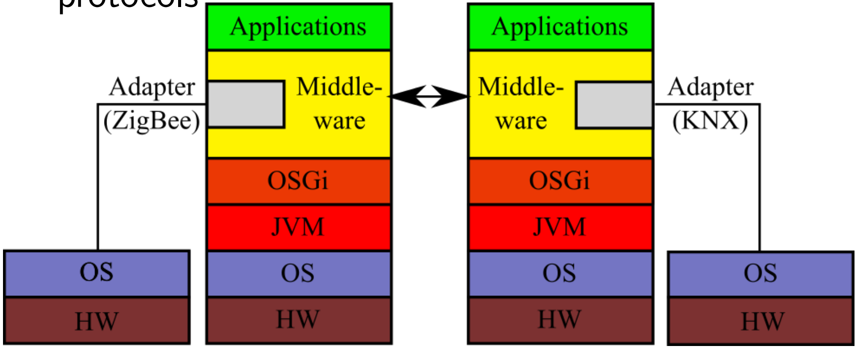
The Solution

- Adapters

Non-OSGi devices (cont'd)

Sensors added as external nodes via adapters

- as other low-computational-power devices
- or devices without JVM
- or devices not supporting the inter-middleware protocols

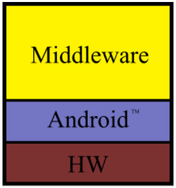


Android ^{TM}

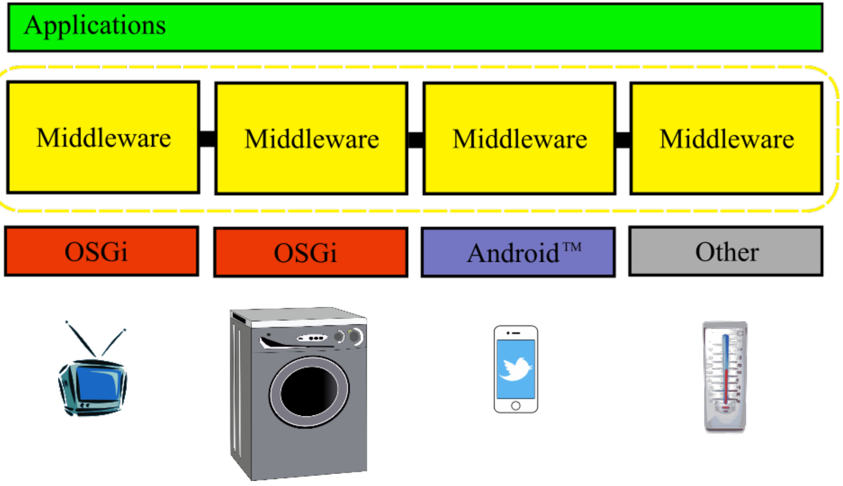
- Operating system, Middleware, and application framework of Google ®.
- Open-source
- Implementations on
  - Cellular phones
  - Netbooks
  - Tablets
  - TV sets

UniversAAL on Android ^{TM}

The UniversAAL middleware can directly be ported to Android ^{TM}.



UniversAAL on any Device



Introductory Example

The Lightning Example

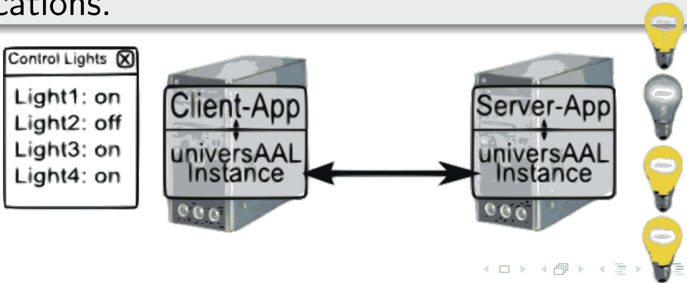
Scenario

- The client-app. makes a request.
- The Service Bus forwards the request to the server-app., and switches the requested light on.
- Real lights can be switched on/off with slight modifications.

Introductionary Example
The Lightning Example

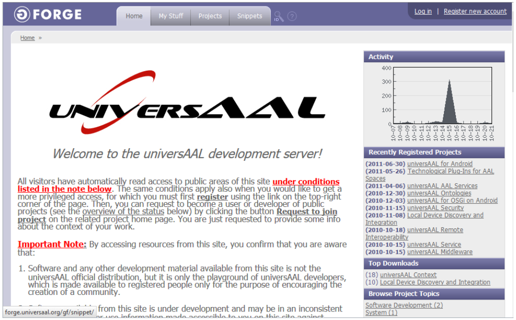
Scenario

- The client-app. makes a request.
- The Service Bus forwards the request to the server-app., and switches the requested light on.
- Real lights can be switched on/off with slight modifications.



Preparation

1. Register at forge.universaal.org/wiki/support:RD_First_Steps



Welcome to the universAAL development server!

All visitors have automatically read access to public areas of this site **under conditions listed in the note below**. The same conditions apply also when you would like to get a more privileged access, for which you must first **register** using the link on the top-right corner of the page. Then, you can request to become a user or developer of public projects (see the overview of the status below) by clicking the button **Request to join project** on the related project home page. You are just requested to provide some info about the context of your work.

Important Note: By accessing resources from this site, you confirm that you are aware that:

1. Software and any other development material available from this site is not the universAAL official distribution, but it is only the playground of universAAL developers, which is made available to registered people only for the purpose of encouraging the creation of a community.

from this site is under development and may be in an inconsistent

forge.universaal.org/gf/snippet/

Recently Registered Projects

(2011-06-30) universAAL for Android
(2011-05-26) Technological Plug-Ins for AAL Spaces
(2011-04-06) universAAL AAL Services
(2010-12-30) universAAL Ontologies
(2010-12-03) universAAL for OSGi on Android
(2010-11-15) universAAL Security
(2010-11-08) Local Device Discovery and Integration
(2010-10-18) universAAL Remote Interoperability
(2010-10-15) universAAL Service
(2010-10-15) universAAL Middleware

Top Downloads

(18) universAAL Context
(10) Local Device Discovery and Integration

Browse Project Topics

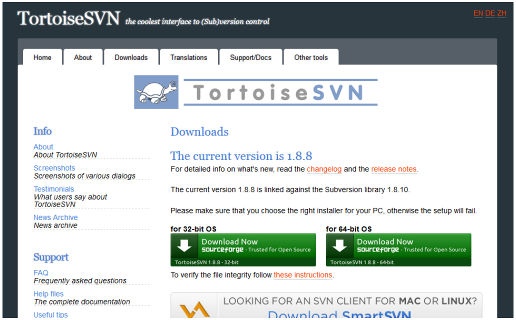
Software Development (2)
System (1)

;

2. From the *Project*-tab, choose and join the groups *Support* and *Ontologies*;

Install Software

3. Apache SubVersioN Client (SVN)


TortoiseSVN the coolest interface to (Sub)version control
EN DE ZH
Home
About
Downloads
Translations
Support/Docs
Other tools
TortoiseSVN
Info
About
About TortoiseSVN
Screenshots
Screenshots of various dialogs
Testimonials
What users say about TortoiseSVN
News Archive
News archive
Support
FAQ
Frequently asked questions
Help files
The complete documentation
Useful tips

Downloads

The current version is 1.8.8

For detailed info on what's new, read the changelog and the release notes.

The current version 1.8.8 is linked against the Subversion library 1.8.10.

Please make sure that you choose the right installer for your PC, otherwise the setup will fail.

for 32-bit OS
Download Now
sourceforge - Trusted for Open Source
TortoiseSVN 1.8.8 - 32-bit
for 64-bit OS
Download Now
sourceforge - Trusted for Open Source
TortoiseSVN 1.8.8 - 64-bit

To verify the file integrity follow these instructions.

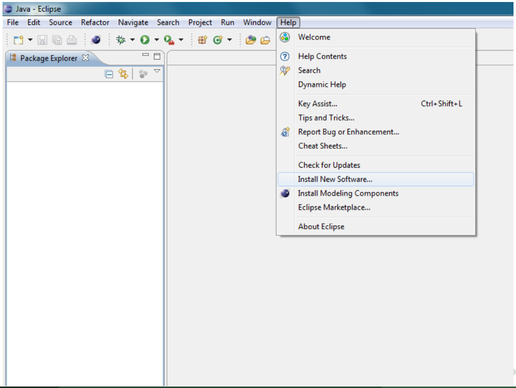
LOOKING FOR AN SVN CLIENT FOR MAC OR LINUX?
Download SmartSVN

Figure: free SVN client at tortoisesvn.net;

4. Check-out from fully-recursive repository forge.universaal.org/svn/support/;

Install Software (cont'd)

5. Java JDK6 (version!);
6. Eclipse (with reference to Java JDK6) ;
7. AAL Studio from http://depot.universAAL.org/eclipse-update

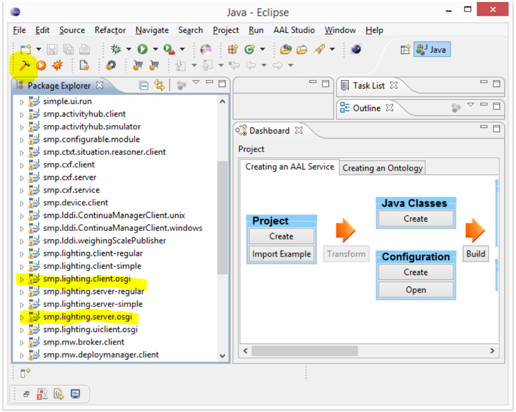


Import the Sources into `Eclipse`

8. Inside the Package Manager, *Import: Maven: Existing Maven projects*;
9. Our samples are `smp.lighting.server.osgi` and `smp.lighting.client.osgi` ;
10. Keep all projects selected!

Compile the Lighting Example

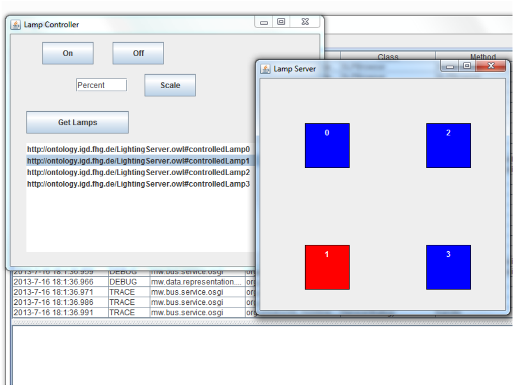
11. From the Package Explorer choose the two projects, and click on the hammer in `AAL Studio`;



simple.ui.run
smp.activityhub.client
smp.activityhub.simulator
smp.configurable.module
smp.ctxt.situation.reasoner.client
smp.cxf.client
smp.cxf.server
smp.cxf.service
smp.device.client
smp.lddi.ContinuaManagerClient.unix
smp.lddi.ContinuaManagerClient.windows
smp.lddi.weighingScalePublisher
smp.lighting.client-regular
smp.lighting.client-simple
smp.lighting.client.osgi
smp.lighting.server-regular
smp.lighting.server-simple
smp.lighting.server.osgi
smp.lighting.uiclient.osgi
smp.mw.broker.client
smp.mw.deploymanager.client

Run the Lighting Example

12. Select tab *Run:Run Configurations*;
13. Choose *Example-Lighting-LATEST_Complete* ;
14. *Run*.



```
http://ontology.igd.fhg.de/LightingServer.owl#controlledLamp0
http://ontology.igd.fhg.de/LightingServer.owl#controlledLamp1
http://ontology.igd.fhg.de/LightingServer.owl#controlledLamp2
http://ontology.igd.fhg.de/LightingServer.owl#controlledLamp3
```

```
2013-7-16 18:1:36.959  DEBUG  mw.bus.service.osgi            or
2013-7-16 18:1:36.966  DEBUG  mw.data.representation....     or
2013-7-16 18:1:36.971  TRACE  mw.bus.service.osgi            or
2013-7-16 18:1:36.986  TRACE  mw.bus.service.osgi            or
2013-7-16 18:1:36.991  TRACE  mw.bus.service.osgi            or
```

References

[1] UniversAAL. Universal open platform and reference specification for ambient assisted living. url = "http://www.universaal.org/index.php/es/about/about-deliverables", 2013. Retrieved on November 3 ,2014.

[2] M. Mosmondor. universAAL: Technical insights. In *AAL Interoperability Days (MACSI 2014)*, European commission, Brussels, Belgium, February 2014.

[3] A. Dey and G. Abowd. Towards a better understanding of context and context awareness. In *in Proc. Workshop on the What, Who, Where, When and How of Context-Awareness, affiliated with the CHI 2000 Conf. on Human Factors in Computing Systems*, The Hague, The Netherlands, April 2000.

[4] M. Debes, A. Lewandowska, and J. Seitz. Definition and Implementation of Context Information. In *in Proc. 2nd Workshop on Positioning, Navigation and Communication & 1st Ultra-Wideband Expert Talk (UET'05)*, 2005.

References (cont'd)

[5] Lars Vogel. OSGi Modularity - Tutorial. url = "http://www.vogella.com/tutorials/OSGi/article.html". Retrieved on November 18, 2014.

[6] P. Kriens. When Applications can Roam Freely. In *Panel of Consumer Communications & Networking Conference 2006 (CCNC 2006)*, January 2006.

[7] H. Cervantes and R. S. Hall. Automating Service Dependency Management in a Service-Oriented Component Model. In *Proc. 6th Workshop on Component-Based Software Engineering*, May 2003.

[8] T. Gruber. Toward Principles for the Design of Ontologies Used For Knowledge Sharing. *Int. Journal Human-Computer Studies*, 43:907–928, November 1995.