

Developer Handbook- 1st EvAAL competition

(Evaluating AAL Systems through Competitive Benchmarking)

<http://evaal.aalooa.org>

V Special theme on Indoor Localization and Tracking

CIAMI Living Lab

25-29 July, 2011

Valencia, ES

Contents

1	INTRODUCTION	3
1.1	ARCHITECTURE	3
1.2	COMPETITION PROCEDURE.....	3
2	REQUISITES.....	5
3	DEMONSTRATOR RUN	6
3.1	DOWNLOAD THE RUNNING ENVIRONMENT.....	6
3.2	IMPORT MAVEN PROJECT	6
3.3	RUNNING	8
3.4	STOPING THE APPLICATION	8
4	DEVELOPMENT PHASE	9
4.1	IMPLEMENTATION.....	9
4.1.1	<i>Download example and template project</i>	<i>9</i>
4.2	UPDATE CONFIGURATION	10
4.2.1	<i>Pax run configuration</i>	<i>10</i>
4.2.2	<i>Running configuration</i>	<i>10</i>
5	ADVANCED TOPICS	11
5.1	COMPILATION	11
5.2	ADVANCED PAX CONFIGURATION	12
5.3	IMPLEMENTING IN OTHER PACKAGES.....	12
5.3.1	<i>Making OSGi Bundle as a solution.....</i>	<i>12</i>
5.4	FAST RUN	13
5.5	IMPLEMENTING A UNIVERSAAL SOLUTION	13

Table of Figures

Figure 1	Competitor's Node Architecture	3
Figure 2	Components to be installed	5
Figure 3	import Project.....	6
Figure 4	Maven project to import.....	7
Figure 5	Imported Project.....	7
Figure 6	Fast access to run command	8
Figure 7	Click procedure to run maven install command	11

1 Introduction

This guide will show the different steps that should be followed to run the competitor application for the competition. First off, competitors will be guided through the setup to install the required software tools which are necessary for developing universAAL applications.

Once the setup is ready competitors will download and import the code. This will enable competitors to run a demonstrator.

Downloading a second piece of code, they will be able to start development by creating a class that implements certain java interface.

Finally the competitor will be guided to configure, run his/her application through the Pax Runner in order to be able to run it properly.

1.1 Architecture

The code for the competition works over universAAL platform, which is eventually running over an Apache Felix OSGi container. To make things easier for competitors an adaptation layer has been added (displayed in blue in Figure 1).

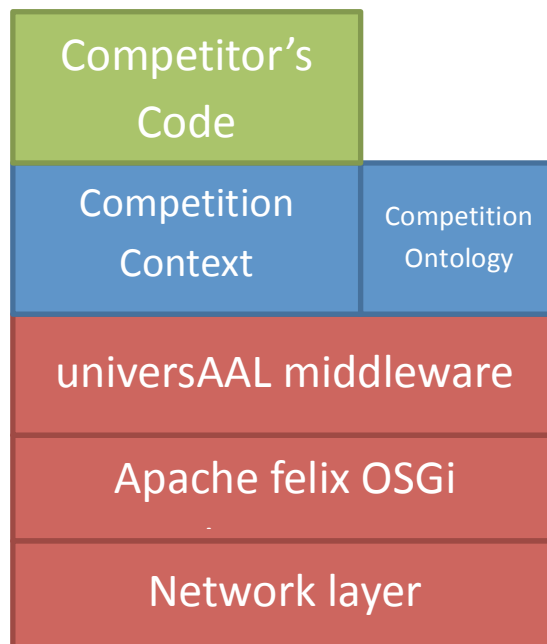


Figure 1 Competitor's Node Architecture

This guide will focus on how to interface with such layer, and how to run an instance in the competitor's node.

1.2 Competition procedure

First off, a platform running universAAL will be configured to listen to all events generated by the competitor, from the beginning to the end of the competition. This machine, connected to the LAN, will be located in a computer room next to living lab.

The competitor should place his/her computer inside the living lab and plug it to the LAN. So all the events could be received and registered by the computer located at the computer room.

Once the competitor demonstrator is configured the localization system will be evaluated in two phases as details “Annex 1 test procedure details” (read it for specific details):

In phase 1 each team must locate a person inside an Area of Interest (Aoi), a place that will be later disclosed.

After that in phase 2 a person that moves inside the Living Lab must be located and tracked. The path followed by the person will be the same for each test, and it will not be disclosed to competitors before the application of the benchmarks.

In this phase each localization system should produce localization data with a frequency of 1 new item of data every half a second (this will be also used to evaluate availability). The path followed by the person will be the same for each test, and it will not be disclosed to competitors before the application of the benchmarks.

In order to see a sample path please read “Annex 1 test procedure details” and the “Refinement of evaluation criteria and procedures”.

2 Requisites

The first step is to install development environment:

1. JDK¹
2. Eclipse 3.5.2² ("Galileo SR2") although Eclipse 3.6.x will also be compatible, as long as the M2Eclipse³ plugging is able to be installed.
3. Together with a SVN client (your preferred choice, for example Tortoise⁴ or any eclipse SVN plugin).

Eclipse will also need certain plug-ins, add the address of the update site to the list of update sites in Eclipse. In version 3.5 and 3.6 of Eclipse, this can be done by first selecting "Install New Software..." under the "Help" menu. The update site is:

<http://depot.universaal.org/eclipse-update/>

You can permanently add this update site to your maven update site list by clicking on the "Add...", the "Name" field could be any name you wish (like for example: "universAAL update site"); the "Location" field must contain the update site url.

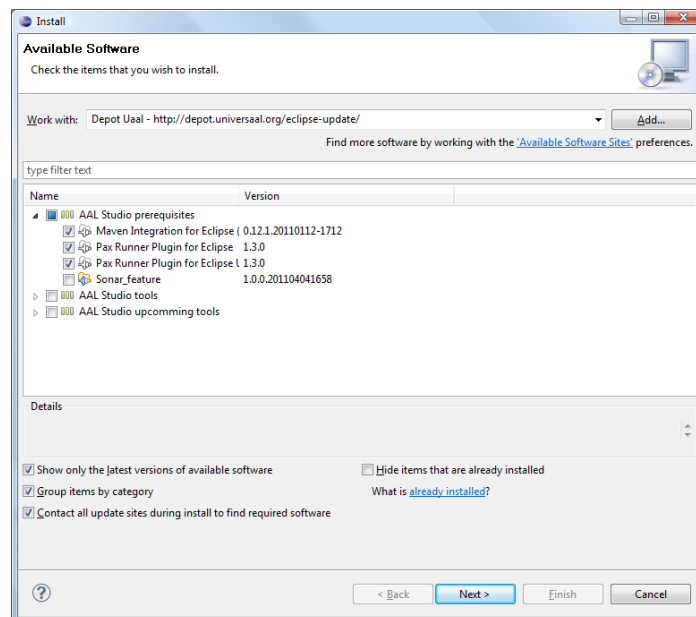


Figure 2 Components to be installed

For the purpose of this competition not all components in the update site need to be installed, just the maven integration for eclipse (M2Eclipse). M2Eclipse can also be downloaded from the home project update Site⁵.

Click "Next" and accept the licences. Then Eclipse will request to restart itself click "yes".

¹ <http://www.oracle.com/technetwork/java/javase/downloads/jdk-6u25-download-346242.html>

² <http://archive.eclipse.org/eclipse/downloads/drops/R-3.5.2-201002111343/index.php>

³ <http://eclipse.org/m2e/>

⁴ <http://tortoisesvn.tigris.org/>

⁵ <http://download.eclipse.org/technology/m2e/milestones/1.0>

3 Demonstrator Run

3.1 Download the running environment

To download the necessary files to run an example (and later the competitors implementation), use the SVN client to “check out” the following URL:
<https://svn.aaloo.org/projects/evaal/Editions/Ed2011/SDK-ev11/location.run>

Create your working copy (local location) in any location like `c:\dev\svn\evaal\`, it is also recommended to download it directly into the eclipse workspace.

3.2 Import Maven project

Once the code is downloaded, to import the code open Eclipse and follow these steps:

1. Go to “File -> Import” (or left click on the package explorer pane and then select “Import”). The following dialog will appear:

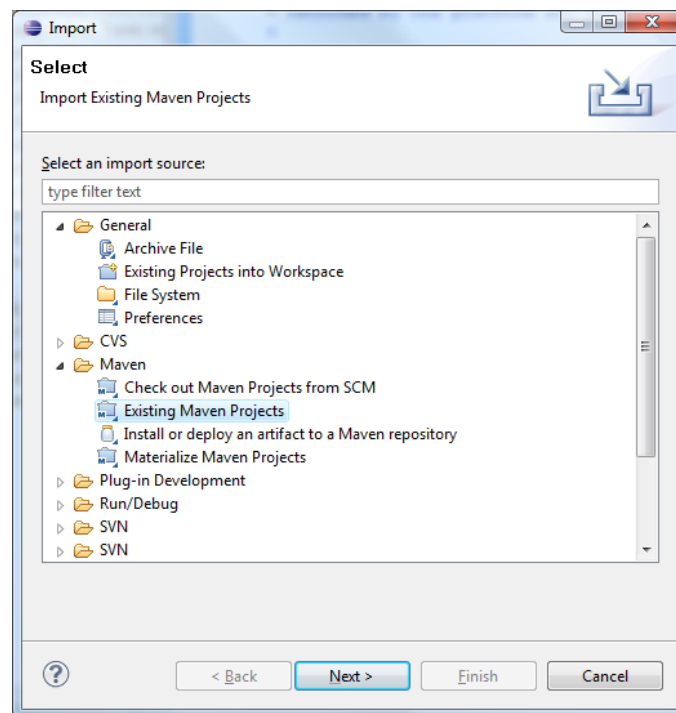


Figure 3 import Project

2. Select Existing Project to import:

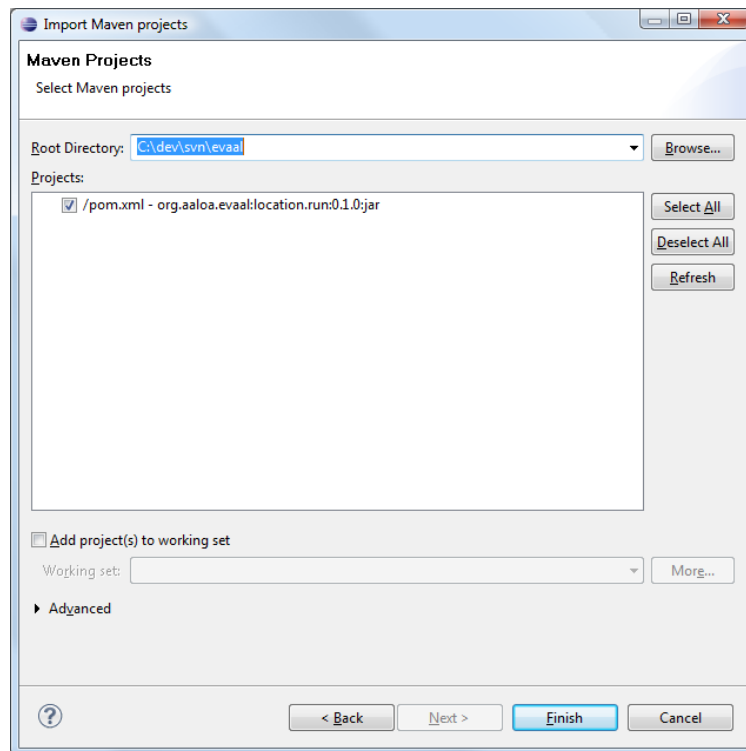


Figure 4 Maven project to import

3. Click finish, you should have now a new project in your package explorer:

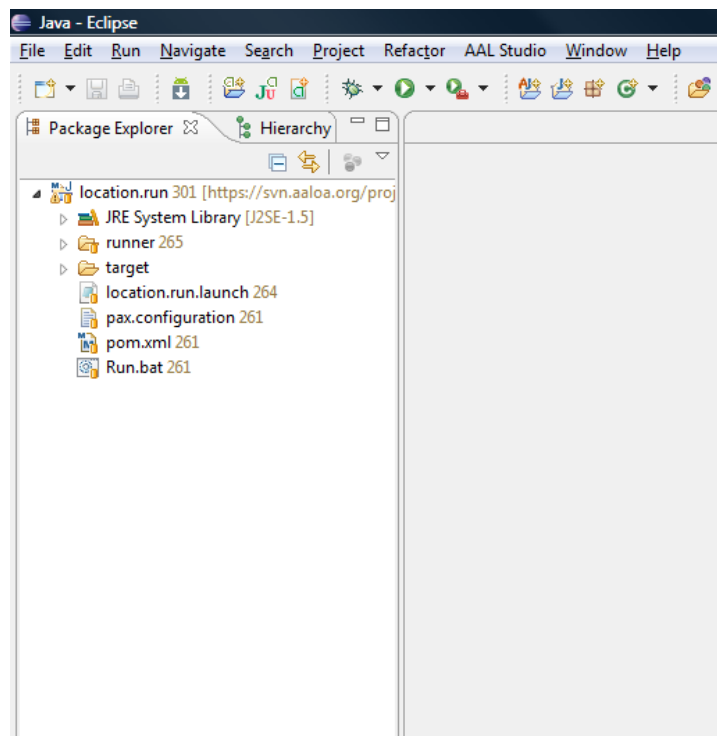


Figure 5 Imported Project

3.3 Running

To run just left click over the location.run project and select “Run as -> "Run as -> Maven build" (see Figure 6).

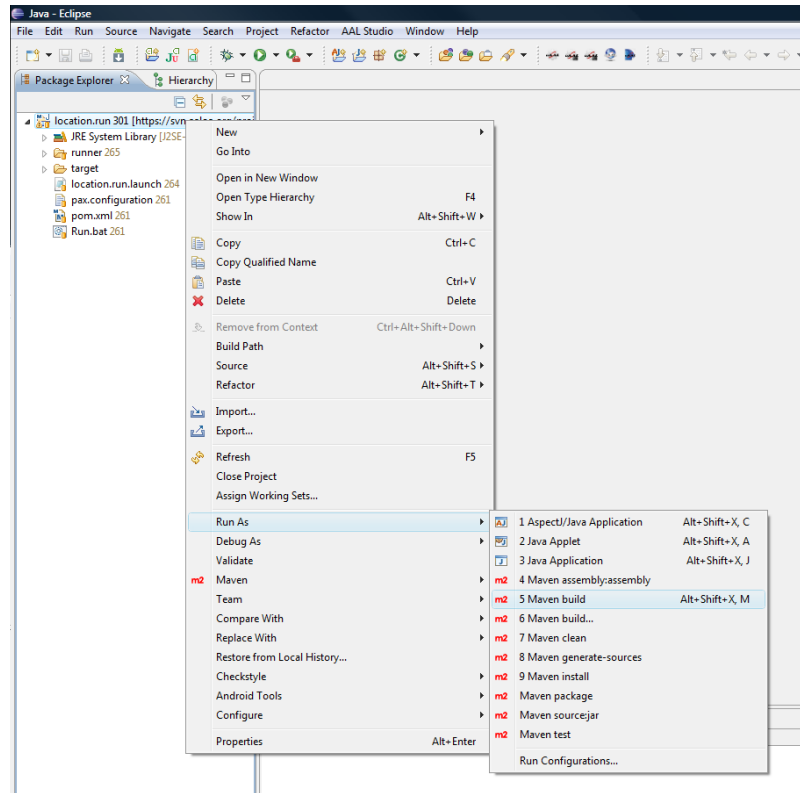


Figure 6 Fast access to run command



This will download all necessary bundles and libraries (check the console), so make sure there is internet connection. This may take some time, specially the first time, so please be patient.

[NOTE]
If you use “Maven build...” (emphasis on the dots), this will generate a new running configuration. It is recommended not to use this option, and if clicked accidentally remove the configuration in the “Run -> Run Configurations...” menu by right clicking on the “Maven Build-> location.run(1)” element.

[NOTE]
If this does not start any running configuration just add the appropriate run configuration.
Using the “Maven build...” (note the dots), on the right click menu over location.run, the new run configuration should have “pax:run” as goal.

3.4 Stopping the application

Once the test is finished, stop the run type “shutdown” in the console.

Clicking on the red square button (), will leave the application running in background, and cause that subsequent runs will use out-dated classes. So **DO NOT** use this button ().

4 Development Phase

4.1 Implementation

An interface is provided for the competitor to implement. This interface (*org.aaloo.evaal.location.competition.Activator.CompetitorInterface*) provides 3 methods to be completed:

- *getCompetitorId()* : must return an identification string for the competitor team.
- *start()* : is the starting point for the competitors code.
- *stop()* : instruct when to stop the competitors code.

When the competitor wishes to publish a location coordinate, he/she should use *org.aaloo.evaal.location.competition.Activator.locate* method.

Any competitor's code must comply with the following:

- Built into one or more .jar libraries/bundles (see 4.2.2 to include them in the running configuration)
- One class should implement this interface:
org.aaloo.evaal.location.competition.Activator.CompetitorInterface.
(remember to configure the run environment, see section 4.2.2, so it is able to locate this class)
- Such class should be allocated in the following package:
org.aaloo.evaal.location.competitor

4.1.1 Download example and template project

For your convenience there is a template maven project. Download and import (just like in sections 3.1 and 3.2) the following project:

<https://svn.aaloo.org/projects/evaal/Editions/Ed2011/SDK-ev11/location.sample>

There you should find a Dummy class and a Template class; both will comply with the implementation requisites for competitor's code. In fact Dummy class is the class executing during the demo run (see section 3), so this should be a good example to understand the development phase.

The template class may also be used to start new development; it provides the skeleton that should be completed. It is recommended to copy the template into a new class, and then develop on top of the new class.

This maven project is ready to be built into the application so it is recommended to develop using this project, see section 5.1 on how to compile it. After compilation the run procedure (see sections 3.3 and 3.4) will import this project by default, so the configuration steps are minimal (just the procedure explained in 4.2.2, if there are no extra libraries to be loaded).

4.2 Update configuration

Before running the developed solution some configuration of the running environment is needed, in this section the configuration needed is explained. Once configured you can run your code just follow sections 3.3 and 3.4 to stop.

4.2.1 Pax run configuration

Edit the file `/location.run/pom.xml` to import the necessary jars and bundles into the run. This pom file is configured to use the pax provisioner plugin ⁶for maven.

The pom.xml file should include a *provision* tag for each jar or bundle needed in the run in the maven-pax-plugin configuration section. Each tag should have the form `<provision>url</provision>`, where the url depends on whether the imported jar is a bundle or not, and whether it is listed in the maven repository or it is stored locally in a file use the following table to include the correct libraries:

	Library	Bundle
Maven File	<code>wrap:mvn:group.id/artifact.id/version</code>	<code>mvn:group.id/artifact.id/version</code>
	<code>wrap:file:/C:/some/directory/file.jar</code>	<code>file:/C:/some/directory/file.jar</code>

These urls are provision commands⁷ used by Pax Runner.

If the location.sample project is used (instead of a new one), and no other jar or bundle are necessary then this configuration should be ready to go, as it already imports the generated jar. Although a compilation (see section 5.1) should be done each time the code changes, in other that the latest version is the one actually running.

4.2.2 Running configuration

By default the project runs a default example code (Dummy) that publishes random locations.

To change this behavior, configure the *CompetitorClass* property to point your own implementation of *CompetitorInterface* in the file:

</location.run/runner/configurations/location.competition/competition.properties>.

This is a required step to make sure your code is correctly run.

⁶ <http://www.ops4j.org/projects/pax/construct/maven-pax-plugin/provision-mojo.html>

⁷ <http://ops4j1.jira.com/wiki/display/paxrunner/Provisioning>

5 Advanced Topics

This section should only be read by experienced developers, or to find solutions that may be compatible with your implementation.

5.1 Compilation

It is very recommended to use maven project management, as all projects supplied are maven projects. If you don't know how to use maven please refer to the maven project home page for more information:

<http://maven.apache.org/>

If you just need to compile a maven project, then maven script install will do it. Once you can see your project in the explorer view press right click and "Run As -> Maven install"

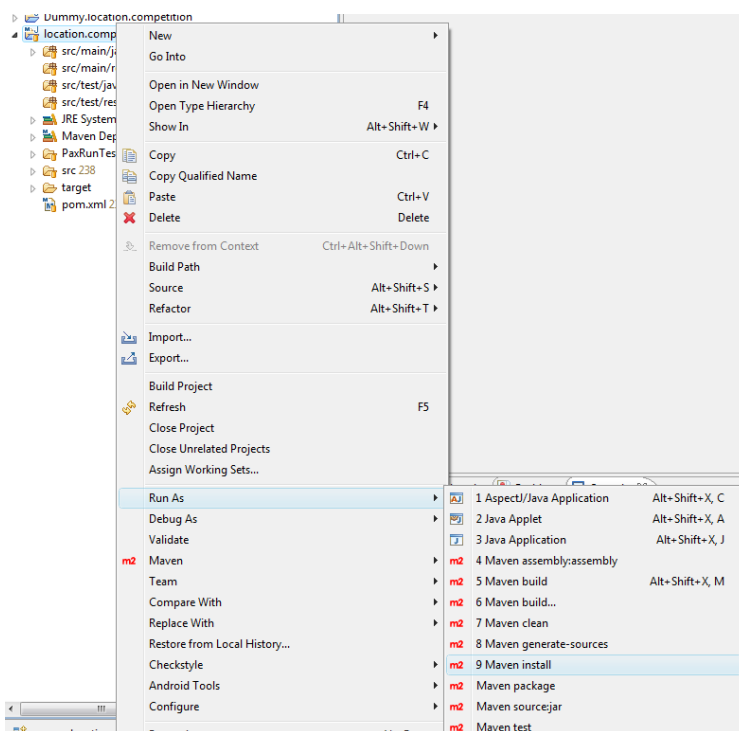


Figure 7 Click procedure to run maven install command

All required platform artifacts should be downloaded (check it on the eclipse console). And the project will be compiled. This step may take a few minutes the first time, please be patient.

[NOTE]

If you get the following error:

....

```
[INFO] Compiling 5 source files to C:\test\location.competition\target\classes
```

```
[INFO] -----
```

```
[ERROR] COMPILATION ERROR :
```

```
[INFO] -----
```

```
[ERROR] Unable to locate the Javac Compiler in:
```

```
C:\Program Files\Java\jre6\.. \lib\tools.jar
```

Please ensure you are using JDK 1.4 or above and not a JRE (the `com.sun.tools.javac.Main` class is required). In most cases you can change the location of your Java installation by setting the `JAVA_HOME` environment variable.

...

Go to “window -> preferences -> java -> installed JRE's” menu, and remove all except the latest JDK.

If there is not any JDK, add one. This is done by clicking on the “Add” button, select “Standard VM”, click “Next”, select “JRE Home” as your JDK local installation folder (it should be something like: `C:\Program Files\Java\jdk1.6`) and click “Finish”

5.2 Advanced PAX configuration

Each artifact may have extra configuration that are appended after the url, one or more configurations may be appended:

- **@<run_level>**: <run_level> is an integer representing the start level of the bundle. If present and OSGi Start Level Service is available then the bundle is installed with that specific start level.
- **@nostart** : present the bundle should not be started. Default, if the value is not present the bundle will be started depending on the service configuration. This configuration can be useful if you do not wish to start the `location.competition` bundle (an there for your code) right away
- **@update**: if present and the bundle was already installed before, the bundle will be updated. The default behavior, if the value is not present, the bundle will be updated or not depending on the service configuration. Useful for those bundles/libraries which will change during development and testing.

These configurations are especially useful to make sure the bundles are updated to their latest version and that they start in the desirable order.

5.3 Implementing in other packages

It is recommended to implement the `CompetitorInterface` in package `org.aaloo.evaal.location.competition`. But if you need to allocate your code elsewhere there is a bit of extra configuration (apart from that on section 4.2).

1. Download and import (just like in sections 3.1 and 3.2) the following project:
<https://svn.aaloo.org/projects/evaal/Editions/Ed2011/SDK-ev11/location.competition>
2. Edit `location.competition/pom.xml` so that the `<Import-Package>` will include your package.
3. Compile the project as explained in 5.1

5.3.1 Making OSGi Bundle as a solution

Another way around allocating the implementation in other package which is not `org.aaloo.evaal.location.competition` is generating an OSGi bundle instead of a library. That is if

you have the knowledge on the OSGi platform and how to build OSGi bundles, if not please refer to the OSGi Alliance⁸ home page, or the Apache Felix⁹ implementation that is used.

If you opt for a Bundle, remember that your publications (calling `org.aaloa.evaal.location.competition.Activator.locate` method) should not start until the `start()` method is called.

Never the less the default configuration will try to load the package, so it is recommended not to delete de location.sample from the pax configuration (see section 4.2.1), so that the package is loaded even though it will not be used. To disable the package loading the `<Import-Package>` tag (see section 5.3) should be deleted.

5.4 Fast run

The running procedure, explained in section 3.3 may take a lot of time each time it is executed. This is because it updates all the bundles, libraries and necessary configuration. But once this is done the environment can be very quickly executed by running the script:

```
location.run/runner/run.bat
```

Remember this run method will not update your bundles or libraries; so you may have to either run as explained in section 3.3 or update manually the jar files. That is why this method is discouraged

To manually update a Bundle just copy it, replacing the old one, in the folder:

```
location.run/runner/bundles
```

5.5 Implementing a universAAL solution

If you are interested in the development over universAAL, or if you already know how to develop over universAAL platform, there is also the possibility of creating your own universAAL context publisher and discard de adaptor. Then you may use the adaptor project as template:

<https://svn.aaloa.org/projects/evaal/Editions/Ed2011/SDK-ev11/location.competition>

Remember to run only one context publisher, so delete the location.competition bundle from the pax configuration (see section 4.2.1).

The format of the context events should have a *Competitor* (all these classes are inside `org.aaloa.evaal.location.ont` Package) as subject and `Competitor.PROP_publishesLocation` as predicate. Remember that this property is a *TimedPoint* so you will need to update this property before publishing each event.

For more development details on how to develop for universAAL please refer to:

<http://depot.universaal.org>

⁸ <http://www.osgi.org/>

⁹ <http://felix.apache.org/>