ADSS – Developer Manual

ASPIRE Decision Support System

Politecnico di Torino

April 11, 2016

Contents

Disclaimer

This manual documents the ASPIRE Decision Support System as released in date April 11, 2016 and its content it is likely to change as new versions are released.

1 Preparing the development environment

The ADSS requires some preparations before a developer can start modifying it. It is mainly written in Java (or analogous technologies), so that it should be OS-independent.

To setup a fully working development environment you need to follow the following initial steps:

1. install the following software into your system:
   - JDK 7.x or superior (`http://www.oracle.com/technetwork/java/javase/downloads/`);
   - SWI-Prolog 7.2.x or superior (`http://www.swi-prolog.org`);
2. install the following software in the ASPIRE virtual machine (or in your system if you want to do a local build):
   - CodeSurfer 2.x or superior (with a proper builder license);
   - the ASPIRE Annotation Extractor;
   - the ACTC;
   - at least one ASPIRE software protection tool.

After that you have two choices. The simplest one is to download an already packaged Eclipse, so that:

- if you are using Windows:
  1. download the all-in-one Eclipse binaries from `x` and decompress it somewhere;
  2. download the ADSS workspace from `y` and decompress it somewhere else;
  3. launch Eclipse and when asked for the workspace directory choose your ADSS workspace location.
- if you are using Linux, wait a bit, someday somebody will put it online.

Alternatively, if you are brave enough, you can setup a working Eclipse and download the source from SVN:

1. download an Eclipse 4.x Mars for RCP and RAP Developers from `http://www.eclipse.org/` and install it;
2. launch Eclipse and install the following plug-ins:
   - from `http://download.eclipse.org/releases/mars` install:
     - Subversive SVN Team Provider;
     - EMF - Eclipse Modeling Framework Xcore SDK;
     - EMF - Eclipse Modeling Framework SDK.
   - from `http://www2.imm.dtu.dk/~ekki/projects/ePNK/indigo/update/` install:
     - ePNK Basic Extensions;
     - ePNK Core;
     - ePNK HLPNGs.
   - download `http://www2.cs.uni-paderborn.de/cs/kindler/research/EPCTools/downloads/epctools-2.0.3.plugin.eclipse-3.1.zip` and unzip it in you Eclipse plug-ins directory.
3. from Eclipse imports all the plug-ins from the SVN repository at `https://aspire-fp7.eu/framework/development/ADSS/` — if asked for an SVN connector you can install SVNKit 1.8.x;
4. create an Enclipse launch configuration using all the plug-ins in the workspace;
5. for Windows only, modify the launch configuration in this way:

- add `-Djava.library.path="C:`
  `Program Files`
  `swipl`
  `bin"` to the VM arguments, specifying where you have installed SWI-Prolog binaries;
- add the `SWI_HOME_DIR` environment variable pointing to your SWI-Prolog directory (e.g. `C:`
  `Program Files`
  `swipl`);
- add the `PATH` environment variable pointing to your SWI-Prolog binaries directory (e.g. `C:`
  `Program Files`
  `swipl`
  `bin`).

Remember to install Java, SWI-Prolog and Eclipse for 64 bit or for 32 bit OSses, without mixing them. Under a Debian/Ubuntu you can easily install the JRE and SWI-Prolog via the following command lines:

```
aptitude install openjdk-8-jre openjdk-8-jre-headless
aptitude install swi-prolog swi-prolog-java swi-prolog-nox
```

2 ADSS plug-ins

The ADSS is split in several plug-ins, that are:

- the `eu.aspire_fp7.adss`, developed by POLITO, contains most of the logic of the ADSS;
- the `eu.aspire_fp7.adss.akb`, developed by POLITO, contains the AKB;
- the `eu.aspire_fp7.adss.akb.ui`, developed by POLITO, contains the AKB editor UI;
- the `eu.aspire_fp7.adss.help`, developed by POLITO, contains the ADSS help (and this manual);
- the `eu.aspire_fp7.adss.ui`, developed by POLITO, contains some common ADSS UI classes;
- the `eu.aspire_fp7.adss.util`, developed by POLITO, contains various utility methods;
- the `it.polito.security.ontologies`, developed by POLITO, contains the ontology API;
- the `org.uel.aspire.wp4.assessment`, developed by UEL, contains the Petri Nets API.

In addition you can also find two other (optional) plug-ins developed by POLITO:

- the `it.polito.security.ontologies.samples` contains some ontology examples;
- the `it.polito.security.ontologies.tools` contains some ontology analysis tools.

3 Accessing and expanding the AKB

Most of the AKB is stored into a central ontology that can be easily accessed via the `ADSS.getModel()` method. It returns a `Model` EMF object that contains all the information used by the ADSS. It can be freely read and modified using a set of getters and setters. You can also perform a low-level access to the ontology by using the `Model.getOntology()` method.

The AKB `Model` class contains the following principal getter methos:

`getOntology()`
: retrieves the ontology itself — use it only if you are 100% sure of what you are doing;

`getApplications()`
: retrieves the list of software applications — this method may be removed in the future;

`getAttackPaths()`
: retrieves the list of detected attack paths;

`getAttackSteps()`
: retrieves the list of detected attack steps;

`getProtections()`
: retrieves the list of supported protections;

`getAttacker()`
: retrieves the attacker profile, containing data such as the expertise level and his budget threshold;

`getAttackerTools()`
: retrieves the list of supported attacker tools;

`getSolutions()`
: retrieves the inferred solutions.

You can access a stable AKB in several points, such as:

- in the `PetriNetsConnector.initialize()` method that it is called before actually calling the SPA tool to perform the Petri nets assessments;
- in the `PetriNetsConnector.compare()` method that it is called when comparing two solutions in order to find the best one.

The developer can also choose to look into the following files for more information about what is inside the AKB:

- the file `owl/akb.owl` in the `eu.aspire_fp7.adss.akb` plug-in. It contains the initial OWL ontology loadead when a new project is created;
- the file `xcore/akb.xcore` in the `eu.aspire_fp7.adss.akb` plug-in. It contains the XCore model used to automatically generate the Java classes of the AKB. In here all the available getters and setters are also documented.

4 Modifying the UI

4.1 Adding new steps to the build process

First, note that adding a new build step requires *only* modifying the UI, so that your additional code can be placed anywhere. Basically, there are two things to do: modify the build all and step-by-step actions. To add a new step:

1. open the `eu.aspire_fp7.adss.akb.ui.editors.OverviewPage` class — all the modifications must be performed here;
2. locate the `createAutomaticBuildSection` method and add the new code to be executed to the listener of the `buildAllHyperlink` class field — this will take care of the 'Build All' hyper-link;
3. create a new `org.eclipse.ui.forms.widgets.ImageHyperlink` object as a new class field — this widget will model the button used to launch the new step only;
4. locate the `createStepByStepBuildSection` method, add the code to allocate the new hyper-link and set-up its listener accordingly — this will take care of the 'Step by Step Build' hyper-link.

4.2 Adding a new page to the AKB editor

This modification is very straightforward. To add a new page:

- implement a new concrete class extending `org.eclipse.ui.forms.editor.FormPage` containing the new custom content;
- open the `eu.aspire_fp7.adss.akb.ui.editors.AKBEditor` class — this class model the AKB editor as a whole;
- modify the `addPages` method in order to add the new page to the AKB editor.