# "*SoEfTraceAnalyzer*": a tool for Exploiting Traceability Uncertainty between Software Architectural Models and Extra-Functional Results

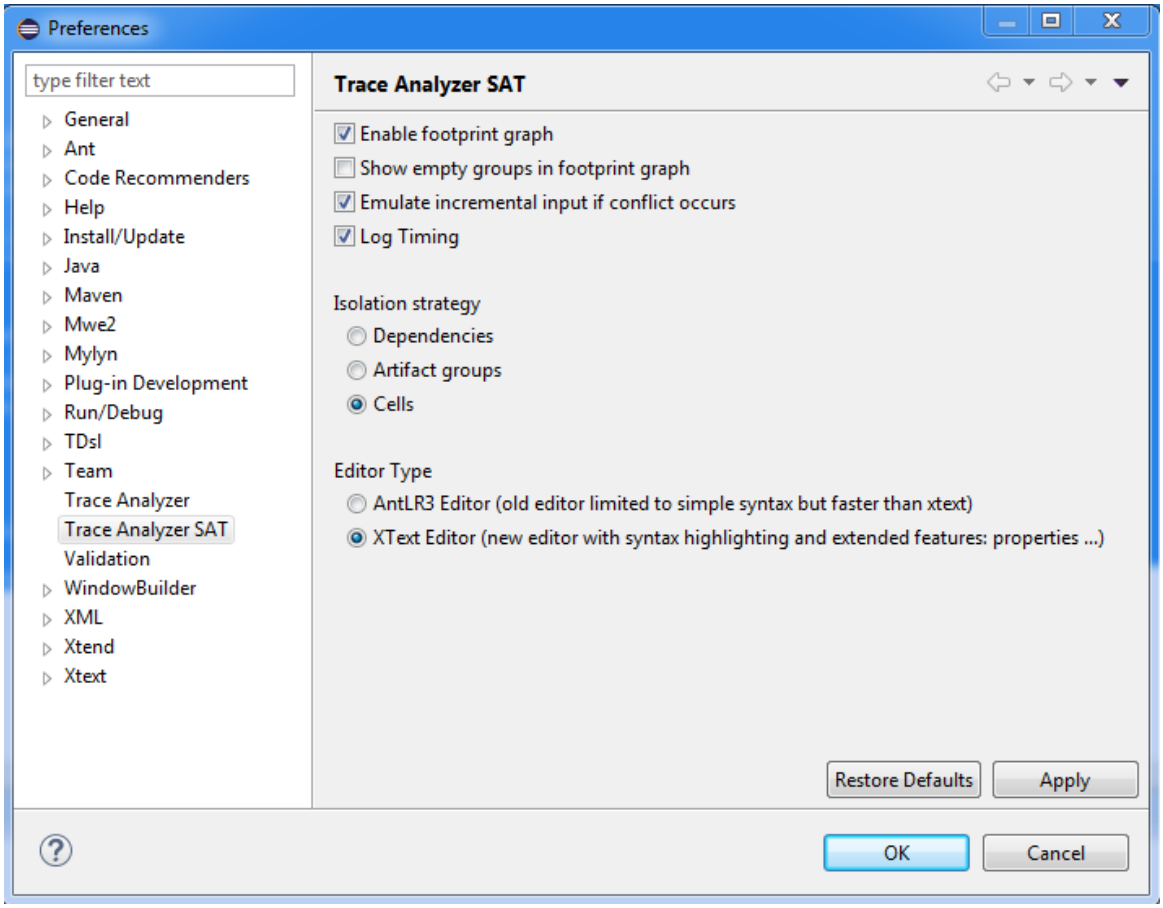
## Installation of the tool

1- Extract the zip file.

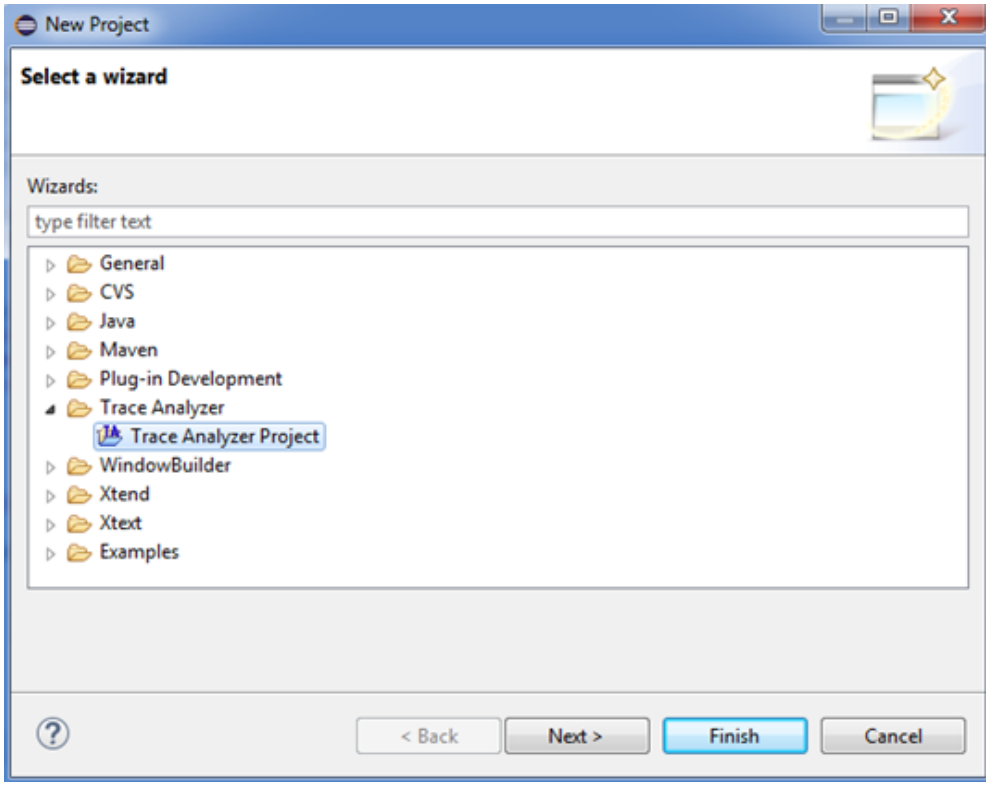
2- Open the file 'eclipse.ini' and setup the path for your Java JDK:

```
-startup
plugins/org.eclipse.equinox.launcher_1.3.0.v20140415-2008.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.win32.win32.x86_64_1.1.200.v20140603-1326
-vm
C:/Program Files/Java/jdk1.7.0_60/bin
-product
org.eclipse.epp.package.rcp.product
--launcher.defaultAction
openFile
--launcher.XXMaxPermSize
512M
-showsplash
org.eclipse.platform
--launcher.XXMaxPermSize
512m
--launcher.defaultAction
openFile
--launcher.appendVmargs
-vmargs
-Dosgi.requiredJavaVersion=1.7
-Xms128m
-Xmx1024m
```

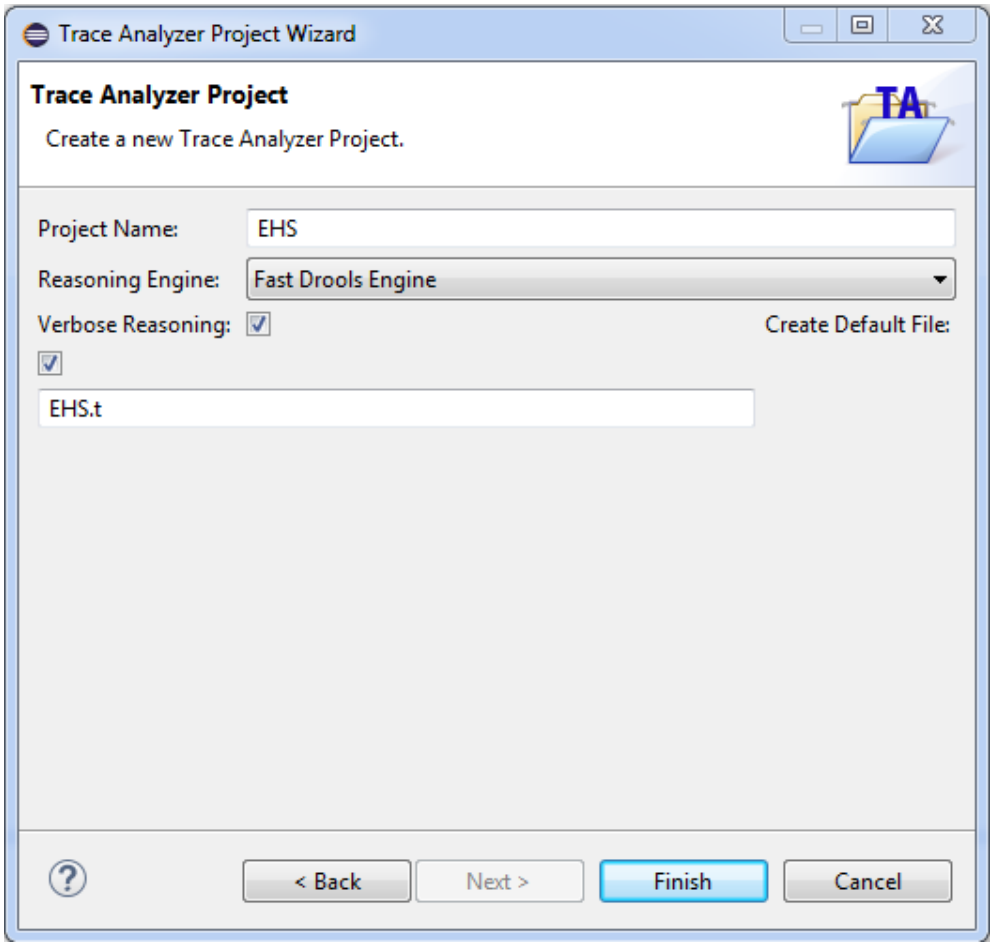
3- Start Eclipse and change the configuration of TraceAnalyzerSAT:



4- Create a new Project:

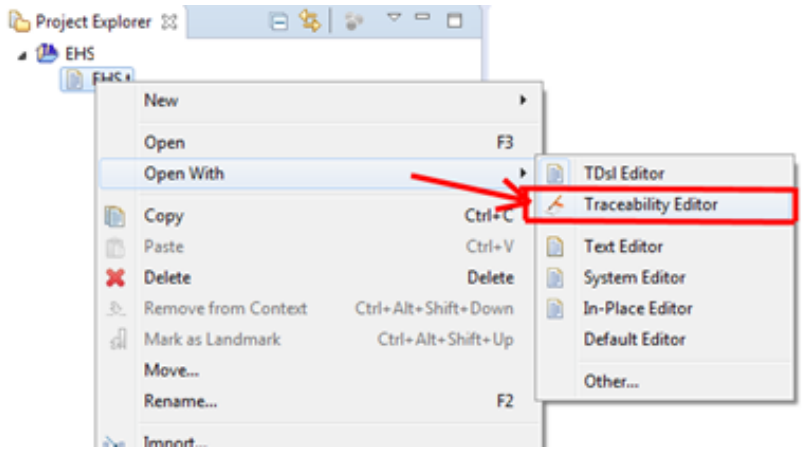



5- Provide a Project Name and select Fast Drools Engine as Reasoning Engine:

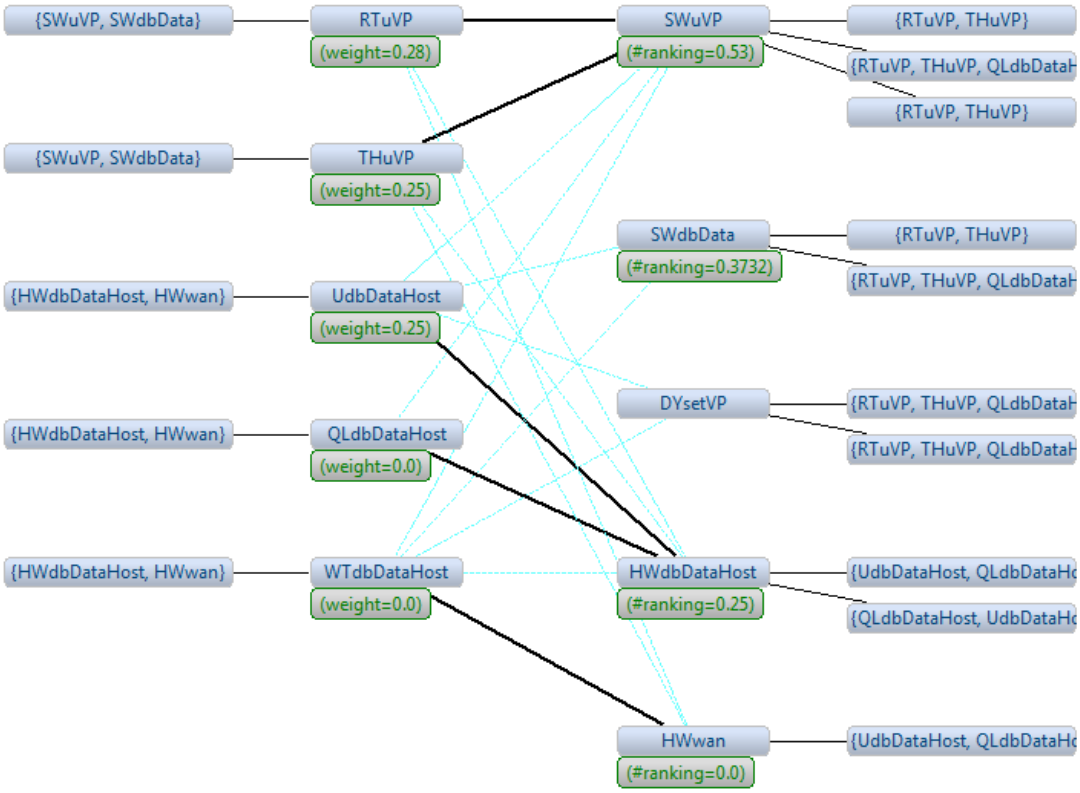


# Running the tool with the E-Health System (EHS) example

1- Open the EHS.t file with the Traceability Editor:




2- Copy the content of the EHS.t file (provided in the zip file) into the file you just created.
3- Save the EHS.t file, close and open it again.
4- The output of the tool is the FootprintGraph of the E-Health System (EHS) example:



## What is new in *SoEfTraceAnalyzer* with respect to TraceAnalyzer[Ghabi-Egyed-Wicsa-2012, Ghabi-Egyed-JSS-2015]:

1. The possibility to **define extra-functional properties** in the input:

```
define visible property weight;
define property guilt;
```

2. The possibility to **provide weight values** for each Result Element (RE):

```
set number weight : RTuVP = 0.28;
set number weight : THuVP = 0.25;
set number weight : UdbDataHost = 0.25;
set number weight : QLdbDataHost = 0.0;
set number weight : WTdbDataHost = 0.0;
```

3. The possibility to **provide guilt values** estimating how much a Model Element (ME) is "responsible" for the corresponding RE to which it is connected. Numbers reported in the following comes from the application of our guilt-based approach [Trubiani-etAl-JSS-2014].

```
set number guilt : RTuVP -> SWdbData = 0.44;
set number guilt : THuVP -> SWdbData = 1.0;
```

4. The possibility to **deduce weight values** for each ME node. Such values are computed by the tool and shown in the Footprint Graph along with the visible properties.

## Illustrative example: Video on Demand (VOD)

STEP1. Definition of **architectural model elements**:

```
perspective ArchModelElements {SW.accessMovies, SW.selectMovies,
SW.loadMovieList};
```

STEP2. Definition of **extra-functional properties**:

```
perspective ExtraFuncProperties {RT.loadMovieList, SEC.accessMovies};


define visible property weight;
define property guilt;
```

STEP3. Definition of **weights and guilt values**:

```
set number weight : RT.loadMovieList = 0.7;
set number guilt : RT.loadMovieList -> SW.accessMovies = 0.4;
set number guilt : RT.loadMovieList -> SW.selectMovies = 0.2;
set number guilt : RT.loadMovieList -> SW.loadMovieList = 1.0;
```

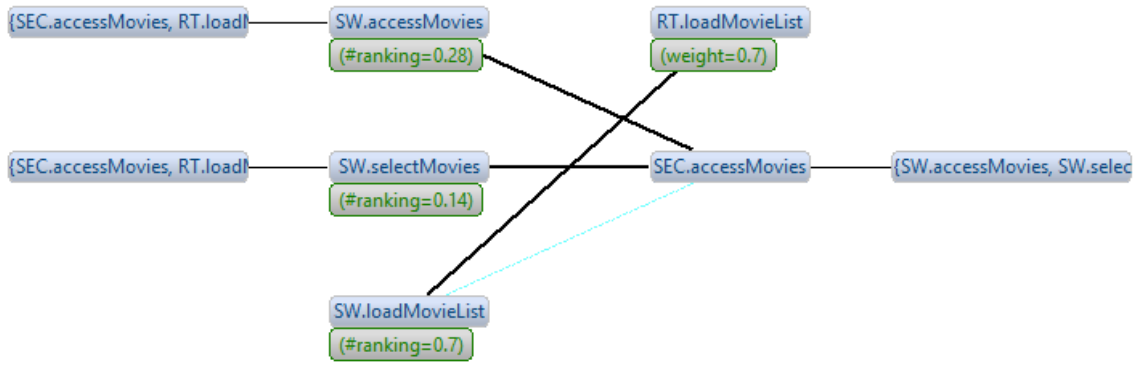
STEP4. Definition of **dependencies**:

```
dependency {SW.accessMovies, SW.selectMovies} atLeast {SEC.accessMovies};

dependency {SW.accessMovies, SW.selectMovies} atMost {SEC.accessMovies,
RT.loadMovieList};

dependency {SW.loadMovieList} exactly {RT.loadMovieList};
```

The output of the tool is the Weighted FootprintGraph of the VOD example:




# References


[Ghabi-Egyed-Wicsa-2012] Achraf Ghabi and Alexander Egyed, “Exploiting Traceability Uncertainty between Architectural Models and Code”, in the Proceedings of Joint Working Conference on Software Architecture and European Conference on Software Architecture (WICSA/ECSA), 2012.

[Trubiani-etAl-JSS-2014] Catia Trubiani and Anne Koziolek and Vittorio Cortellessa and Ralf Reussner, “Guilt-based handling of software performance antipatterns in palladio architectural models”, in the Journal of Systems and Software (JSS), volume 95, pp.141-165, 2014.

[Trubiani-Ghabi-Egyed-ECSA-2015] Catia Trubiani, Achraf Ghabi, and Alexander Egyed: “Exploiting Traceability Uncertainty Between Software Architectural Models and Performance Analysis Results”, in the Proceedings of European Conference on Software Architectures (ECSA), 2015.

[Ghabi-Egyed-JSS-2015] Achraf Ghabi and Alexander Egyed, “Exploiting traceability uncertainty among artifacts and code”, in the Journal of Systems and Software (JSS), 2015.