ASE TEAM 3 & 4

USER MANUAL

HUSACCT –ANALYSE JAVA & C-SHARP

June 23, 2013

NIELS HUBREGTSE – TWAN VERBOOM

CONTENTS

INTRODUCTION

In this document you can find how our group styling should be used. Do not change headings or whatever that is defined in this document.

INTRODUCTION

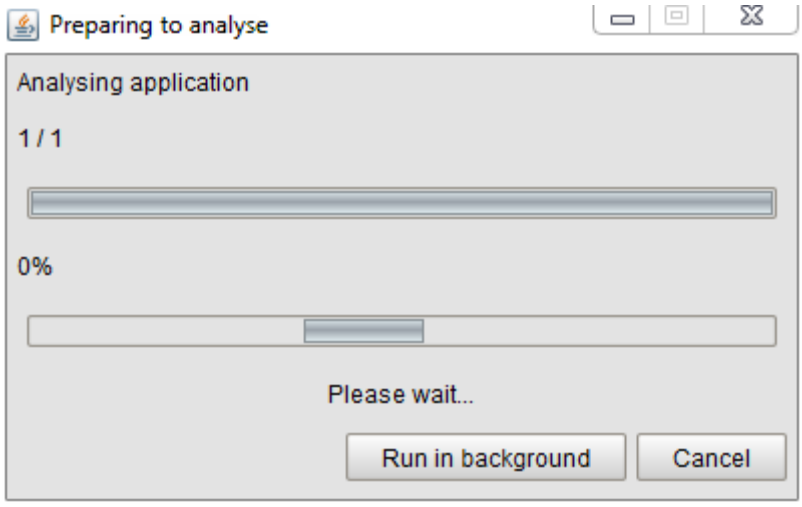
The analyse-service is divided into two major parts that analyse language specific code. These parts do the work of sifting through the code and creating analysable data for the HUSACCT. They work by analysing the code with Antlr.

C# AND JAVA

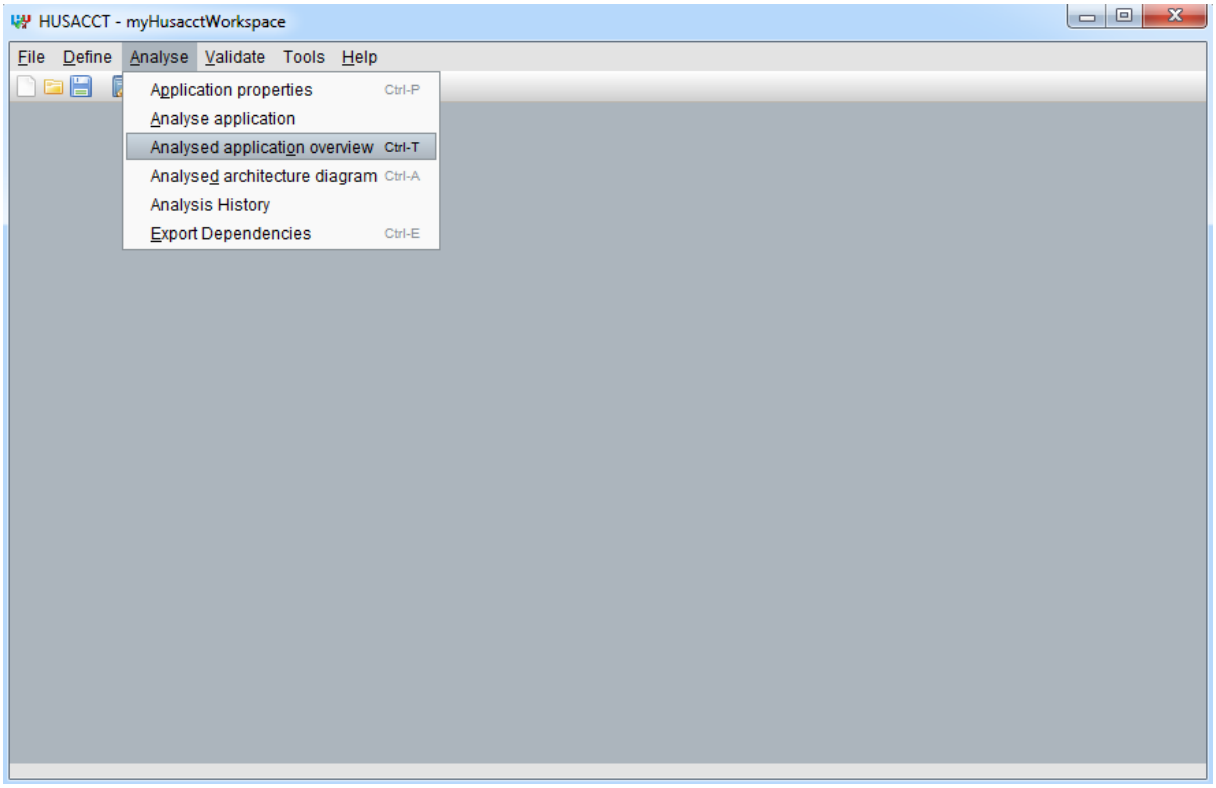
Currently there are two languages supported, those languages are Microsoft's C# (up to version 4.0) and Oracle's Java (up to Java 1.6). Most of the service is based on the same code only the code which really analyses the code itself is different per language. By sharing as much code as possible within the service it became very easy to add a new language. Between C# and Java aren't many differences in reported dependencies or types, but the C# analyser currently supports lambda's and struts already which will be implemented in java 8 and therefore aren't supported by Java yet.

USER INTERFACE

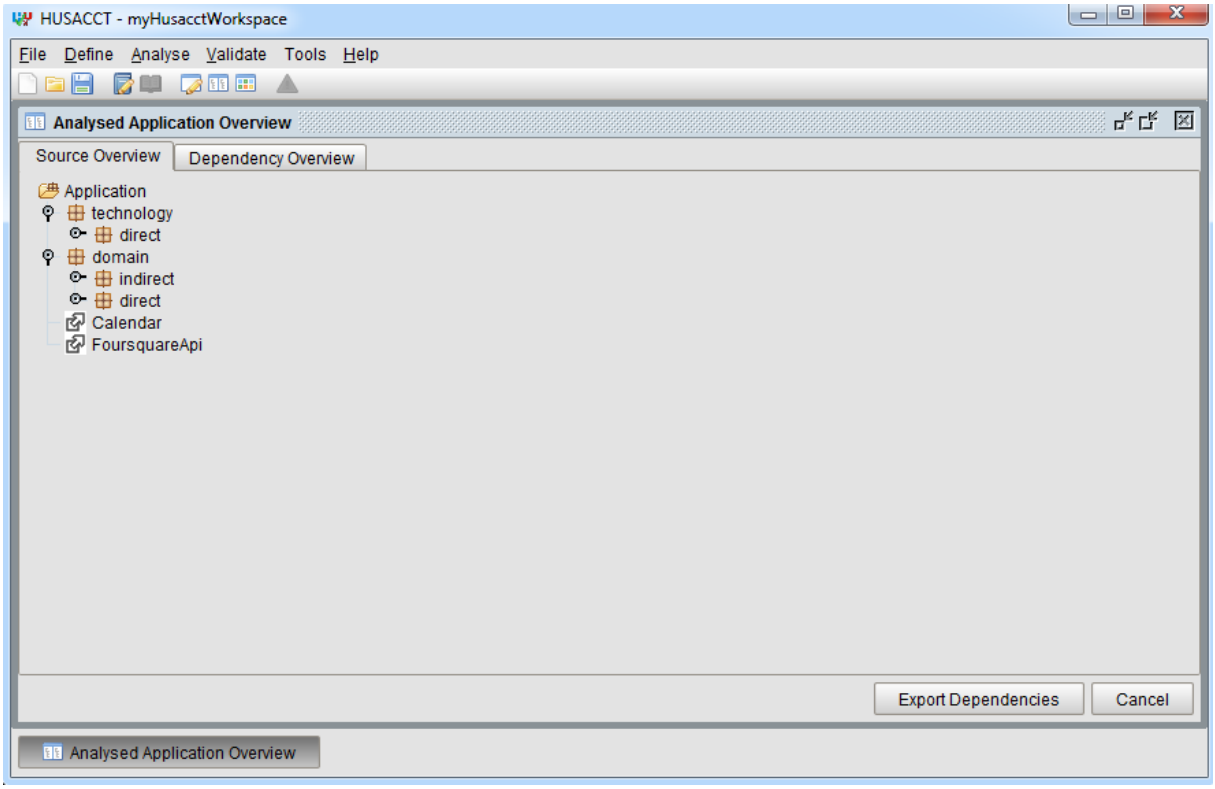
When you run HUSACCT and create a new workspace you have the option to directly analyse a project. There is no difference between direct analysing and analysing on a later moment, when analysing is started the previous analysed information is cleared and the code will be completely reanalysed. Analysing a project can take up some time. When the project contains many files it might take a long time, a small project is analysed within seconds in general. After the project is analysed the system will start building a cache for the dependencies. This can on large projects take longer than the analysing itself. Because the dependencies between files can be complex and lay very deep it takes time to build the cache. On the other hand, when the cache is completely built there will be no waiting while retrieving information about dependencies between two classes. Only with huge projects retrieving information from the cache can take up to about 2 seconds. During this time the interface might freeze but it was either waiting the same time it takes the cache to build or the two second wait time.



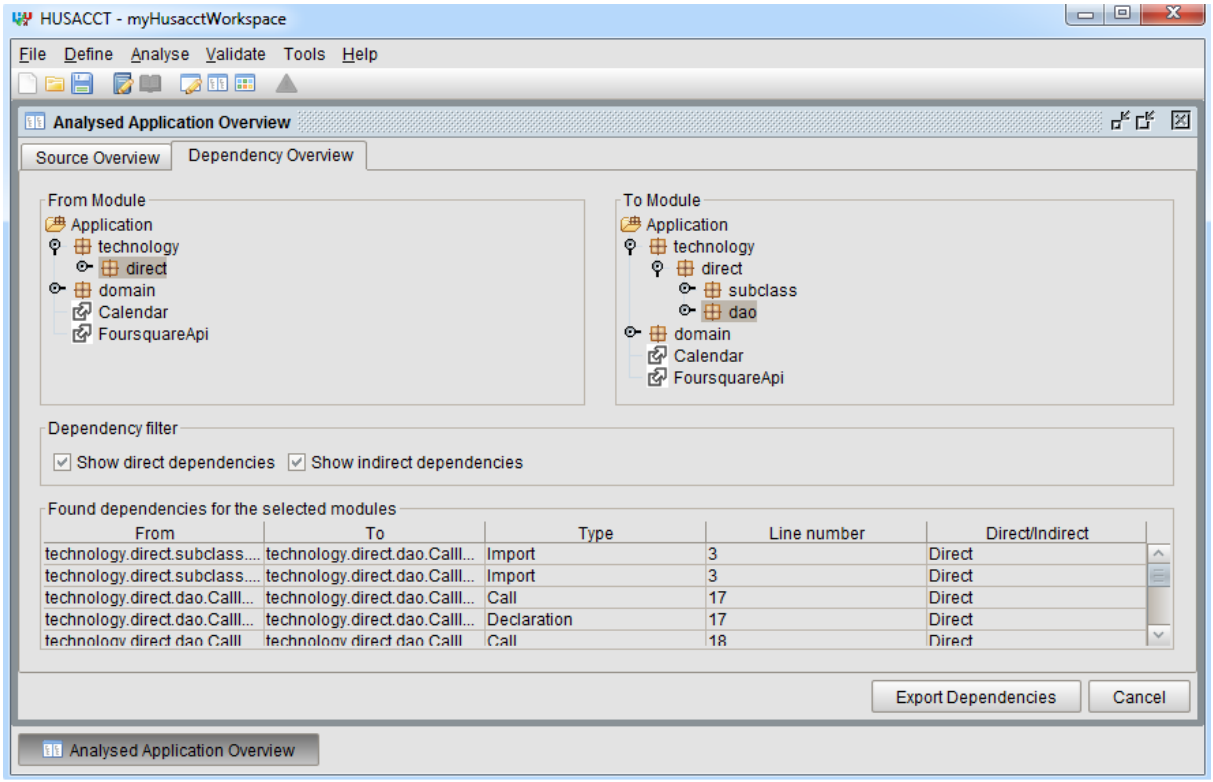
In the above screenshot you notice the two loading bars which display the progress of the analysing. The top bar is implemented for future extensibility for analysing multiple projects, the lower bar is for the progress of analysing the current project. After the project is analysed the lower bar will be running from left to right and back, during this time the cache is being built. Because there is no possibility to give an indication how far the cache is.
After the analysing is finished, the “Analysed Application Overview” is accessible from “Analyse” at the top of HUSACCT.



Within the “Analysed Application Overview” you can do two major things, look at the source tree of your project and look at the dependency overview of your application.
The “Source Overview” is designed to let you browse your application, through the modules and external systems.



The “Dependency Overview” has some more features. Here you can select modules on the left and right to see which dependencies run from one to the other.



If there are dependencies between these modules you can see those in the bottom part of the screen with detailed information about that dependency.
The HUSACCT features a code viewer to directly inspect the code from which this dependency was found. This can be accessed by double click on the dependency in the “Found dependencies for the selected modules” part.