Include v Understand Qualitative:

With a sample size of 382 over the total population of Understand (35%), Include (1%) and the overlap of the two (64%) it was prudent to determine the unique differences to each dependency. Looking at 133 unique samples from Understand, 244 samples from their overlap of dependencies, but only taking 1% of samples for Include would not be sufficient to find the reason for distinct dependencies thus we will take 10% of total samples and delegate them from overlap samples making Include samples 38 and 206 overlap samples. This will give a good idea of the quality that each tool has and determine which is the better tool to use. We will now look at some examples of the majority results found from the samples above. The blue file path will represent the dependent and the red file path will represent the depended.

Overlap:

Example (1): …\hadoop\hdfs\server\namenode\**FSDirSymlinkOp.java** 🡪 …\hadoop\fs\permission\**PermissionStatus**.java

In this dependency it was found that the dependent imports the depended and that the depended is an abstract data type PermissionStatus and is used as so in the dependent:

*PermissionSatus dirPerms;*

*…*

*dirPerms.getUserName;*

Thus here we see that both Include and Understand have found import statements, which that is what Include only does, but Understand also finds method parameters and method calls. This is just one example of the samples that we took here are just a few more that exhibit the same or similar dependency qualities found.

Example (2): …\hadoop\fs\**AbstractFileSystem.java** 🡪

…\hadoop\util\**Progressable.java**

Example (3): …\hadoop\mapred\**TestYARNRunner.java** 🡪

…\hadoop\yarn\api\protocolrecords\**GetApplicationReportResponse.java**

Example (4): …\hadoop\hdfs\protocolPB\**ClientNamenodeProtocolTranslatorPB.java** 🡪

…\hadoop\hdfs\inotify\**EventBatchList.java**

It was found in fact that all samples show these same qualities that an import of the depended is found in the dependent file which shows both Understand and Include both find import statements and Understand found it as a dependency as well because there was a use of the depended as an abstract data type, method calls, variable use, as an interface for another class, inheritance, implementations and others that the Understand tool seeks for as described earlier in the paper. Now we will look at some examples that only the Understand tool found.

Understand Only:

Example (1): ...\java\org\apache\hadoop\security\**TestNetgroupCache.java** 🡪

…\main\java\org\apache\hadoop\security\**NetgroupCache.java**

Here we have a unique to Understand dependency found from the samples. There is no import statement which we know is the only way that Include would find a dependency. Understand finds this dependency because the dependent uses a method from the depended, the reason there is no import for this particular dependency is because the files are in the same directory so the import is not needed.

Example (2): …\hadoop-common\src\test\java\org\apache\hadoop\crypto\**TestCryptoCodec.java** 🡪

…\hadoop-common\src\main\java\org\apache\hadoop\crypto\**OpensslCipher.java**

Here is another example of a unique to Understand dependency found from the samples but slightly different. In this case again there is no import of the depended which clarifies why the Include tool did not find the dependency. The reason the Understand tool found it was because the dependent uses the test package that the depended is in but not directly, the dependent also uses a child of the depended’ method thus there is a dependency in two more ways that Include would not find, imports of packages as they don’t directly include the filename of the depended and the use of a child’s method.

There are many cases as described above which Understand uniquely finds. All of them don’t include an import but have a dependency that Understand finds through the import of a package that the dependent belongs too, an import and use of a child of the depended or singularly or a combination of the former but the depended belongs to the same package/directory as the dependent allowing use without an import. Next we will look at some examples of dependencies only found by the Include tool.

Include Only:

…\hadoop\yarn\server\nodemanager\containermanager\container\**ContainerImpl.java** 🡪

…\hadoop\yarn\server\nodemanager\containermanager\application\ **ApplicationContainerFinishedEvent.java**

This is a case where only the Include found a dependency. The depended is imported which is how Include finds dependencies, but Understand also find these. An interesting thing to note is that although this dependency was not outputted to the csv file it is in the Understand Dependency Browser, curiously the depended is used as so:

*…*

*eventHandler.handle(new ApplicationContainerFinishedEvent(containerId))*

*…*

It directly uses a constructor of the class as a parameter. It is possible that Understand does not find these unique dependencies, although finding it in the Understand Dependency Browser contradicts this statement indicating an error and affecting the precision of the Understand tool. Here are a couple other examples of the same situation described:

Example (2): …\hadoop\yarn\server\applicationhistoryservice\**ApplicationHistoryWriter.java** 🡪

…\hadoop\yarn\server\applicationhistoryservice\records\

**ApplicationAttemptFinishData.java**

Example (3): …\hadoop\yarn\applications\unmanagedamlauncher\

**TestUnmanagedAMLauncher.java** 🡪

…\hadoop\yarn\client\**ClientRMProxy.java**

We have to note that all the samples explored exhibited the situation described earlier so quantitatively we can estimate that all of the 889 unique dependencies found by Include are either an error on our part of the use of Understand or a error in the precision of the tool. Now we will see the qualitative precision and recall of both tools against each other:

Include - Precision = 40404/41293 ~ 0.98

Recall = 40404/63123 ~ 0.64

\*Note that recall would normally use only Include acquired samples

Understand - Precision = 62234/63123 ~ 0.99

\*Note that recall cannot be considered fully as errors cannot by guaranteed and this precision is based on comparison population and not acquired by Understand only population