Disclaimer: Dr. West, THANKS for your consideration in responding to questions and addressing them conscientiously. I am hoping you will honor my request to not grade the original submission (pg. 3-8) and instead to grade to newer one on project "jxmlutil" for reasons relating to build difficulties with the original project – ODK Aggregate (or base on combined effort from both if that is a better fit) -CC

Testing and Build Tooling Assessment: "Jxmlutil" project

The software project on testing analysis I assessed was found on the Google Code projects website. This project revealed several interesting concepts in teaching me about project models, code structuring and modularization. Prior to this project's effort, I was largely ignorant of processes and utilities available in the automated build tooling realm, if I had heard of them. I had known of ant and maven and have configured them on Windows 7 before, built their hello world projects to observe their basic usage (for learning purposes but not for more constructive purposes, such as exploring in context of an actual large-scale project). I learned Maven is somewhat like ANT and it presumably has all of ANT's features plus more, but I am still not quite convinced Maven is as straightforward and simple to use as ANT in practical settings.

To describe the build of my project, I used Subversion to clone its codebase off Google and used knowledge gained after combing through a website literature for Maven/m2eclipse and from exploring with a previously attempted effort that failed – which was ODK Aggregate. While I found trial-and-error to be effective as any other learning enabler, when trying to build and test the final project (jxmlutil), I easily applied build experience acquired to successfully build out this small project (~2 KLOC). There are no instructions for building the project – just a .pom file, but there was only one change made to achieve a build success – I changed the xerces 'GroupID' from ApacheXerces to just 'Xerces' and voila!

To build the project using "maven install", as submitted in the .zip attachment, in my environment, I was able to generate Sonar (static code analysis using a dynamic- presumably JVM-based application accessible @localhost) and Cobertura (.html-)reports, as well as to run the GUI. Though there were no explicitly created unit or integration tests for the project, I created precursory ones under /src/test/java (as is expected by the build tool) for demonstration.

Although, the build process works as far as "maven clean install sonar:sonar cobertura:cobertura," completing all build phases without error, the level of automation saw improvement over the failed ODK Aggregate failed effort. There is no inconsistency between downloaded jars and the versions maven finds in the repositories/expects, so changes to the pom.xml file were sufficient and do not require /.m2/settings.xml to be changed, as did the ODK Aggregrate project, to fix in accordance with guidance on maven's website, under the: 'dependencyManagementMechanism' initiative.

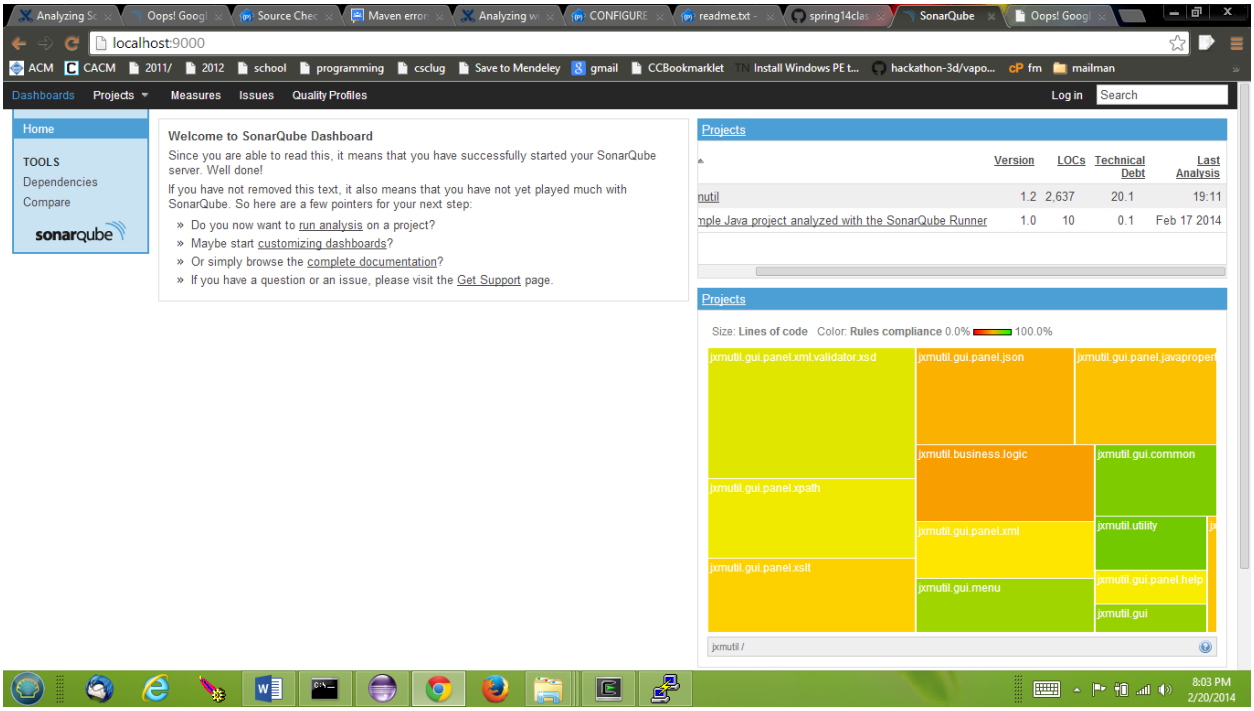
As a 'outgoing note': my level of experience with MAVEN and ANT plugins is still on the lower end, and all the above occurred first on a Windows 8 environment, then I ran it on an EC2 cloud instance of Ubuntu (all of which worked except sonar). I have successfully executed the unit tests and built and visited localhost in the browser on a Ubuntu machine to verify it compiles and runs correctly.

TESTING REPORT:

I used sonar for static code analysis, as suggested, and arrived at the following metrics, as evidenced by the screenshot and rules.csv (included as an in-document object), below:

rules.csv

Metric	Measure
LOC	2637
Methods	71
Classes	32 (32 original; 8 test classes I added in src/test were not included in this metric)
Documentation	68.7%



For test coverage analysis, I used cobertura and obtained the attached 'site' .html files:

cobertura-site-html.zip

PLEASE DISREGARD THE BELOW PROJECT INFO:

Testing and Build Tooling Assessment: FAILED/Request non-graded: "ODK Aggregate"

The testing analysis of the software project, on google code's website, revealed several interesting points useful in teaching me about project models, code structuring and modularization. Prior to this phase of the project's effort, I was largely ignorant of processes and utilities available in the automated build tooling realm, if I had heard of them. I had known of ant and maven and have configured them on Windows 7 before, build their hello world projects to observe their basic usage for learning purposes but not for more constructive purposes, such as exploring in context of an actual large-scale project. I learned Maven is somewhat like ANT and it presumably has all of ANT's features plus more, but I am still unaware whether Maven can achieve exactly what ANT can, entirely, and whether I can improve the projects usage of it in an automated building context, by the end.

To describe the build of my project, I initially used Mercurial to clone the project's codebase and followed extensive instructions after combing through a reasonable amount of website literature but found trial-and-error to be about as effective as any other learning strategy, when trying to understand the build and testing structure/process for my project. Two documents from the project code were of some assistance to me in setting up the build and test environment: CONFIGURE.txt (explains setting up a build environment in Eclipse IDE – see appendix (a)) and README.txt (explains pre-reqs for getting MAVEN to build/compile, regardless of using an IDE). To build the project using "maven install", three steps are required, first, exclusive of whether the developer elects unit testing/IT based on the: MySQL, PostgreSQL, or GoogleAppEngine-Datastore database backend/endpoint:

1) The user must obtain the necessary .jar file for the Database/-Connection (ie:MySQLconnectorJ.jar)
2) A preliminary build inside the "odk-gae-settings" subproject must be completed using ant and the generated .jar moved inside the "eclipse-aggregate-gae\war\WEB-INF\lib".
3) Inside src/main/libs folder, using the README.txt commands, additionally libraries must be added so maven is aware of them (This is the current instruction from the README file…)

Although, the build process has worked for me, to this point, at least as far as "maven install" has successfully completed all Phases of the build order, ie: without FAILURE, the level of automation associated with the build seems amenable to improvement. For starters there may be some inconsistency between downloaded jars and the versioning that maven looks for, so it is possible the settings.xml file could be updated, according to rules on maven's website, under the: 'dependencyManagementMechanism' initiative.

As a 'DISCLAIMER': going into this effort AND as of this point, still, my level of experience with MAVEN, ANT, Maven2Eclipse plugin is minimal at best, and all the above occurred first on a Windows 8 environment, as the project's primary developer/engineer uses Windows environment exclusively. The project seems to lend itself to Linux(/Ubuntu distro) testing and development, but I have not successfully executed the unit tests/IT *ON EITHER* environment, though, I am able to build and visit localhost in the browser of either to verify the sources/project compiles and runs successfully.

To explain the test processes in place, aside from the 3 database backend options available thereto, there are automated testing facilities available through maven and associated EMMA plugin to some degree. The code analyzed primarily serves to accept and validate XForm submissions to the system, which uses a java-based application server (Tomcat or GAE – see appendix (b)) to store the form into a database, so it mostly tests the Server/Upload functionalities and uses JUnit for that. Barring, GAE's DataStoreImpl.java class is excluded from testing per Line#1264 of pom.xml

Appendix: Additional considerations:

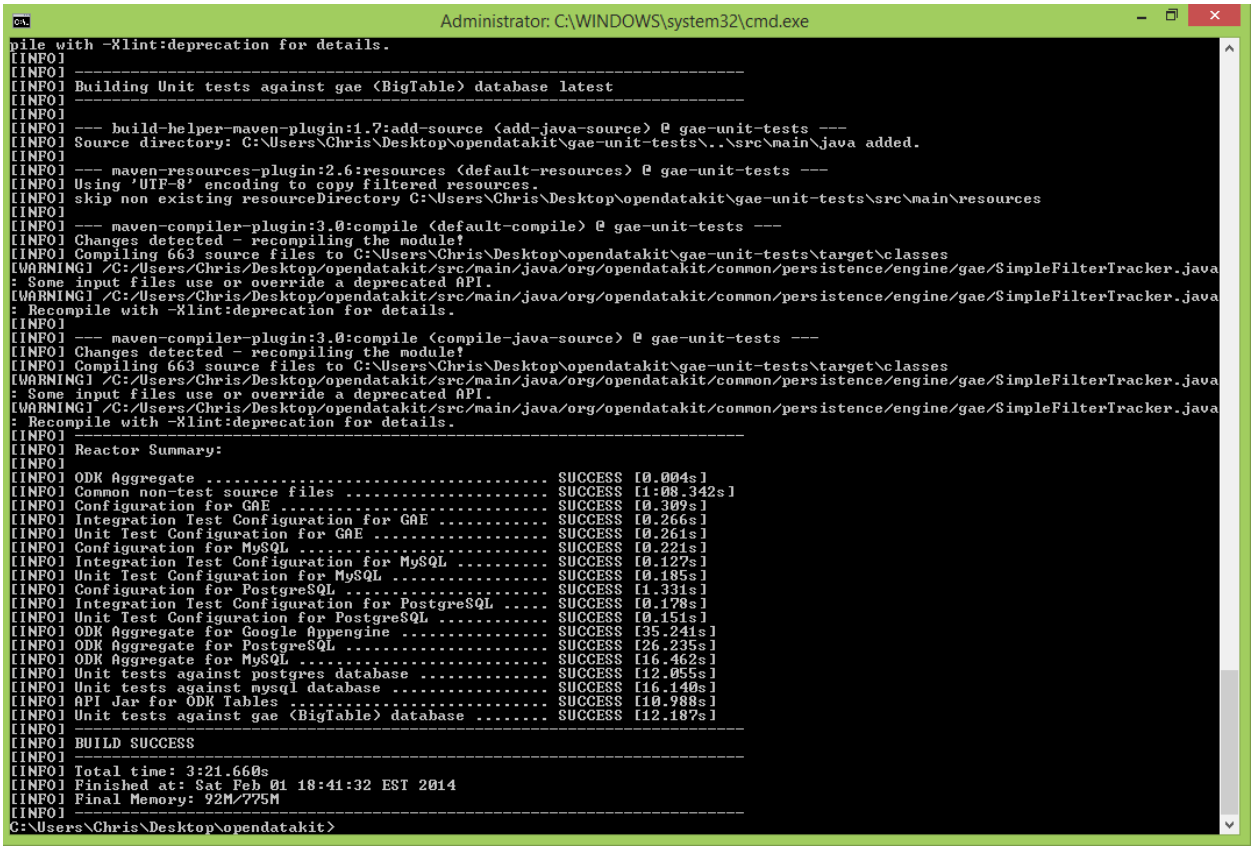
(a) Interestingly, at line 362 (point #12) in the CONFIGURE file referenced above, there is mention of using the Maven plugin for Eclipse (m2e) yet also that support for it is 'dodgy in Eclipse,' so not surprisingly, I did not have much luck building to any level of understandability or practicality using this Eclipse utility. However, I did take an IDE/GUI approach, building the odk-gae-settings subproject using ANT in Eclipse, which then involves 'mov'ing the target/ .jar file from there to a sister directory, in order to build the overall project and run a local(-host) instance of the application from the generated overall-project .jar.

(b) I downloaded several .JARs, which google provides to replicate its AppEngine (GAE) cloud-based server-datastorage environment, experimenting and learning about the testing environments associated to it (vs. the Tomcat backend) and walked through a 'GAE tests' tutorial to analyze the AppEngine context.

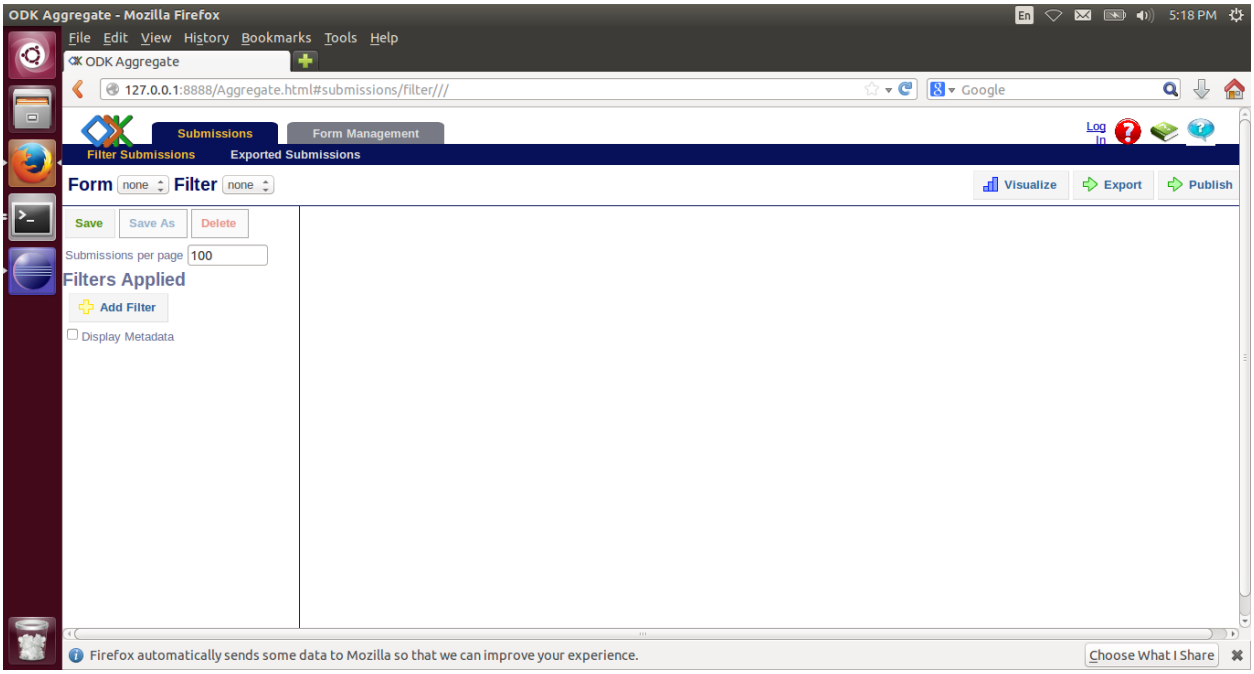
REPORTS AND CODE COVERAGE

Surefire reporting module is used to generate/cover integration tests. However, instead of accommodating the Cobertura plugin for 'testing -code coverage', this project makes use of the EMMA maven plugin from mojo.codehaus.org,declared on lines 940-960 of TOP-LEVEL/pom.xml; and, whereas maven is always capable of producing Cobertura reports, says the default "mvn site"-produced literature on EMMA's website, EMMA produced reports indicate coverage at the: METHOD and BLOCK levels, in addition to Cobertura's CLASS-, LINE- levels and EMMA is reportedly faster than Cobertura (per Sonar site).

This "mvn install" command was successful on Windows and Ubuntu at some point:



Yet, I definitely successfully ran ODK Aggregate from within Eclipse,using their build instructions:


ODK Aggregate - Mozilla Firefox
5:18 PM
File Edit View History Bookmarks Tools Help
ODK Aggregate
127.0.0.1:8888/Aggregate.html#submissions/filter///
Google
Submissions
Form Management
Log In
Filter Submissions
Exported Submissions
Form none Filter none
Visualize
Export
Publish
Save
Save As
Delete
Submissions per page 100
Filters Applied
Add Filter
Display Metadata
Firefox automatically sends some data to Mozilla so that we can improve your experience.
Choose What I Share