**April Monthly Report**

**Conor Griffin**

**X00111602**

For the purpose of this report I will focus on reviewing the 4 weeks of April, weeks 10, 11, 12 and 13 of my Internship programme at the Department Of Social Protection. I am currently working and have been since week 3 the development of a new Continuous Integration platform 2.0. This involves the development of a new git-based platform and “nugetifying” components which involves the use of Nuget package manager and server to store components (projects, clusters, files, dll’s) in Nuget packages. This project when finished will result in a new architecture to for continuous integration and exponentially decrease the time automated tests take on both the BOMi2 and BOMi4 applications.

Week 10 – Worked on CI project

I continued working on the team project in the development of a new CI platform 2.0, I worked on "nugetifying" two components from BOMi's application, a .csproj called corejava and a .dll called nakedobjects.net. Working on the corejava component after numerous errors we realised that it depends on the nakedobjects.net .dll, so I had to "nugetify" that first. So far all the components we have put into nuget packages have been c# projects (.csproj files), we haven't "nugetified" any other type of file, the nakedobjects.net is a .dll file so in order to "nugetify" this component I had to edit the build PowerShell scripts Build.ps1. This scrips calls a function: Build-NugetSolution -Package, this compiles using a artifact.yml file and looks for a .sln file to build the solution and put it into a nuget package, however since this component is a dll it does not have a .sln file because there is no source code to compile, all we wanted to do was call the artifact.yml file and zip the .dll into a nuget package. To do this i created a new function to the build.ps1 script called Build-NugetFromDll, it gets called when you have specified in the artifact.yml file that you are "nugetifying" a .dll and not a c# project file with a .sln. I then had to edit the Pipeline.ps1 script so that when creating the pipeline for the nuget package it had to know whether the component was a .sln or a .dll so it would know what function to call (Build-NugetSolution or Build-NugetFromDll). After successful testing on my local machine of the edits to the scripts and new functions I then committed and pushed my changes to the scripts repository (BuildScripts) and now these changes are updated for everyone to work off.

Week 11 – Continued to work on the development of new CI platform project

I worked on nugetifying the nakedobjects dll and another component called iTextSharp. iTextSharp is used as a PDF library that allows you to create, adapt, inspect, and maintain documents in the PDF format. Both components are Dll files so the process of nugetifying was slightly different. The week before I added function to the build script in powershell to build a nuget package from a dll rather than a solution file with csproject files. I called this function and the compilation of both of these components was successful. I then published the components to the nuget server. Once that was complete I could concentrate on the Build-AllInOne pipelline on Jenkins. After fixing a couple of issue the Build-AllInOne pipelline was successful and both components were complete.

Week 12 – Completed the “nugetification” of multiple components

I worked on 3 components. iTextsharp which was a bare dll, it already existed on the nuget server, all i had to do was remove the dll from the lib folder in Trunk, and update the references to now reference the packaged dll on the nuget server. Once that was done I tested the build all in one to see if there were any compilation errors. Once passed I raised a PR to merge to master branch and it was complete. I also worked on a testing framework component called fitnesse2specflow, this isnt referenced by any other component in the Trunk folder so it was easy to nugetify, remove from trunk and merge to master. The last component was corejava, i have been working on this component for the last 2 weeks, there have been quite a few issues with it because it is a java file (corejava.vjsproj) it is different than any other component/project which are csproj files because most of the codebase is written in C#. I am still currently working on corejava and will hopefully have it complete and nugetified by the end of the week.

Week 13 – Issues with project

I have been working on putting bare dll's into nuget packages. The reason for this was because there are a couple of c sharp projects that use the dll’s as dependencies, compiling these solutions would throw errors because it couldn’t find the dll’s. By putting these dll’s onto our nuget server it would allow the solutions when compiling to look to the server where we would store them and compile successfully without errors.

The issue with putting the dll’s into packages was the codebase for the build all in one (SDM build) has applications (c sharp projects) that depend on different versions of the same dll. We had to package each version and of the dll which wasn’t the issue, the issue was that when we were updating project references to now look at a different hint path of our packaged dll it would update every csproj and vbproj file to look at the same version of the dll rather than look at their own version. For example we had multiple versions of the dll nunit.framework.dll, each version used by different application in SDM. We use a script called Update.ps1 that has a function Update-NugetRefs that runs through SDM\Trunk folders and updates every csproj, vbprob’s hint paths to look at the newly nugetified packaged version on the server. Example..

<Hint Path> ..\..\packages\nunit.framework.2.4.7\lib\net35\nunit.framework.dll</Hint Path>

Unfortunately the script doesn’t care for different versions of the dll and will update everything to reference the same packaged version of the dll. This is something I am looking into and hope to fix the script before we go any further.