**Oregon State University**

*New Build/Distribution Structure for Apache OpenOffice*

Computer Science Capstone Project

2013 - 2014

Team Members:

Colin Bradford

Nicholas Smith

Matthew Wilson

Client:

Jan Iverson

Mentor:

Steve Hathaway

**Table of Contents**

Introduction

Original Requirements Document

Updated Requirements

Weekly Reports (P3s)

Final Poster

Project Documentation

How we learned new technology

What we learned

Appendix 1: Code

Appendix 2: Photos etc.

**1. Introduction**

This project was requested by Jan Iverson from Apache. It was requested because the current Apache OpenOffice build system for the Windows environment was outdated, complicated, and scared away potential contributors. The project reduces the complexity of the current build system, which should attract more Windows developers. Jan Iverson was our client, and Steve Hathaway was a local mentor. Both Jan and Steve mostly just supervised. They helped our better understand the current build system, and the Visual Studio project files. Our team consisted of Colin Bradford, Nicholas Smith, Matthew Wilson. Colin Bradford maintained project documentation. Both Nicholas and Matthew developed the conversion script. All three worked together to convert modules.

**2. Original Requirements Document**

**Team Name:** Walnut Crusade

**Team Members:**

Matt Wilson – wilsomat@onid.oregonstate.edu

Colin Bradford – bradfoco@onid.orst.edu

Nicholas Smith – smitnich@onid.oregonstate.edu

**Client:**

Jan Iversen - [jani@apache.org](mailto:jani@apache.org), Apache Software Foundation

**Additional Assistance:**

Steve Hathaway – shathaway@apache.org, Apache Software Foundation

**Introduction to the problem:** Building the OpenOffice software through Visual Studio as a Windows developer is difficult and has to be done through CygWin with makefiles. The idea is to move from a makefile based platform to a Visual Studio based platform. If possible, we would also allow the easy conversion from this platform to other platforms.

**Project Description:** Make it easy to build the system in Visual Studio and then convert between different build systems.

**Requirements:**

* Create a Visual Studio vcxproj file for every makefile in the AOO suite
* Create a Visual Studio solution file encapsulating all converted modules
* Ensure AOO solution can be built with the old build system using some modules built using Visual Studio to facilitate proper integration
* Allow AOO suite to be built from within Visual Studio without any additional tools beyond what is already required
* Allow for conversion from vcxproj files into makefiles using XML parsing/stylesheet applications, or other build platform formats

**Versions:**

* Version 0.1 is manually converting one module from the current build system into a vcxproj file, and testing it to be sure it builds when placed within the current build system
* Version 0.2 is converting 10 modules into vcxproj files and testing their results within the current build system
* Version 1.0 is writing a script that will help streamline the conversion process between modules and vcxproj files. Note that this version is optional, and may not be feasible. If it isn’t feasible, future module conversion will be done manually like in versions 0.1-0.2.
* Version 1.1 is converting 35 modules into vcxproj files.
* Version 1.2 is converting 60 modules into vcxproj files.
* Version 1.3 is converting 85 modules into vcxproj files.
* Version 2.0 is placing all converted vcxproj files into a Visual Studio solution file and cleaning up the solution for future work

**Design:**

* One-to-one relationship between modules in the current AOO build system to (XML) vcxproj files that can be built within Visual Studio
* Periodically the projects should be built and then inserted into the old AOO build system, whereby we can verify that the entire suite builds using some modules built with makefiles and other modules built with vcxproj files
* One Visual Studio solution file containing all converted vcxproj files
* Create a script using Apache Xerces to convert makefiles, possibly only parts of them or certain types, into equivalent vcxproj files to help streamline the conversion process
* Apache Xerces parsing of Visual Studio (XML) vcxproj file into other build formats, such as makefiles

**Specific tasks to be undertaken:**

* Build AOO system through CygWin, as is the current procedure
* Conduct research on vcxproj file format, makefile format, similarities & differences, Visual Studio environment variables, build rules, and so on
* Select a module with no dependencies from the AOO solution and convert its makefile into an equivalent vcxproj file, then build this through Visual Studio. Once built, insert this into the current AOO build system and verify that it builds properly.
* Select several more modules, possibly some with dependencies on already converted modules, and repeat the previous task with them.
* Research Apache Xerces functionality, use it to try to write conversion protocol between makefiles & vcxproj files. Don’t spend too much time on this task.
* Convert a lot of makefiles into vcxproj files. Break this up into milestones (25 at a time) to keep it manageable.
* Compose a Visual Studio solution file encapsulating all converted modules.
* Write up documentation on what we’ve learned, how to include new modules in the Visual Studio solution, how to use Xerces, etc.

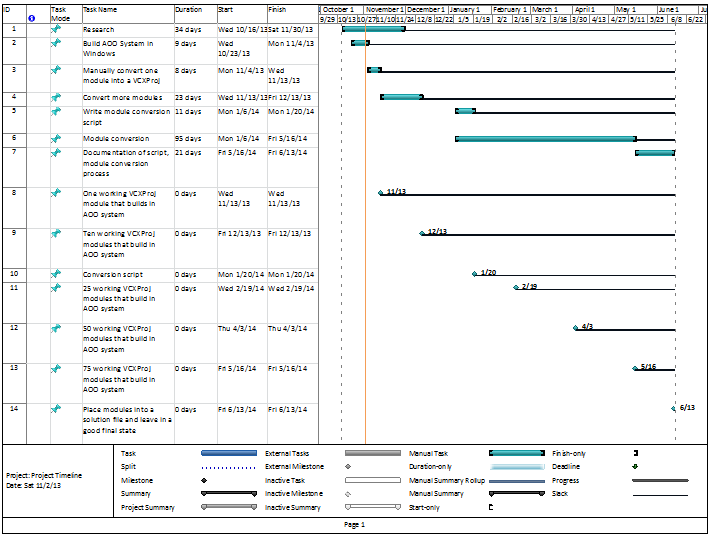
**Risk Assessment:**

1. If there is no feasible way to automate certain aspects of converting between modules & vcxproj files, then we’ll have to do some conversion manually. This will take slightly more time, but should not prevent us from converting a majority of modules.
2. If it takes longer than expected to convert modules, then we won’t be able to hit some of our milestones, but we’ll at least have a theoretical base that can be utilized in the future.

**Testing:**

* The existing build system should be able to build using modules that were built with Visual Studio. This will ensure that the old build system will still work while constructing the new one.
* Possibly try building the Visual Studio solution using modules built with the current build system.
* Make sure that the compiled AOO .exe runs, and that its functionality works as expected
* There are a few modules whose functionality will need to be verified within the built application itself, but most we will assume are fine if the solution is built

**Preliminary Timetable:**



**Roles of the different team members:**

Matt Wilson - Scheduling, P3, Module Conversion

Nick Smith - Module Conversion

Colin Bradford - Secretary, Module Conversion

**Integration Plan:**

* Make sure current build system works when using modules created with Visual Studio projects
* Make sure Visual Studio solution works when using modules created with current build system
* Make sure converted modules do not conflict with each other

**User interface requirements:** None

**References:**

1. Current build system description: [http://wiki.openoffice.org/wiki/Documentation/Building\_Guide\_AOO/Step\_by\_step](http://www.google.com/url?q=http%3A%2F%2Fwiki.openoffice.org%2Fwiki%2FDocumentation%2FBuilding_Guide_AOO%2FStep_by_step&sa=D&sntz=1&usg=AFQjCNH1DGcGPCLaxbFARH8qs6Nmp0RcnA)
2. Apache Open Office website: [http://www.openof](http://www.openoffice.org/)[f](http://www.google.com/url?q=http%3A%2F%2Fwww.openoffice.org%2F&sa=D&sntz=1&usg=AFQjCNF3XRqb7rGTcKc1H3AXaDliRCwI7w)[ice.org/](http://www.openoffice.org/)
3. First attempt at remaking the build system:
   1. Zen of gbuild: <http://wiki.openoffice.org/wiki/Build_Environment_Effort/Zen_of_gbuild>
   2. Gbuild Bootstrapping: <http://wiki.openoffice.org/wiki/Build_Environment_Effort/Gbuild_Bootstrapping>
   3. Module Migration: <http://wiki.openoffice.org/wiki/Build_Environment_Effort/Module_Migration>
4. Time Converter: <http://www.worldtimebuddy.com/?pl=1&lid=5720727,100&h=5720727>
5. Wiki for project: <http://wiki.openoffice.org/wiki/Build_System_Analysis:capstone2013_windows_build>
6. Build System Analysis: <http://wiki.openoffice.org/wiki/Build_System_Analysis>
7. Apache Xerces website: <http://xerces.apache.org/>
8. W3 XML tutorials: <http://www.w3schools.com/xml/default.asp>
9. Visual Studio changes: <http://msdn.microsoft.com/en-us/library/ee862524.aspx>

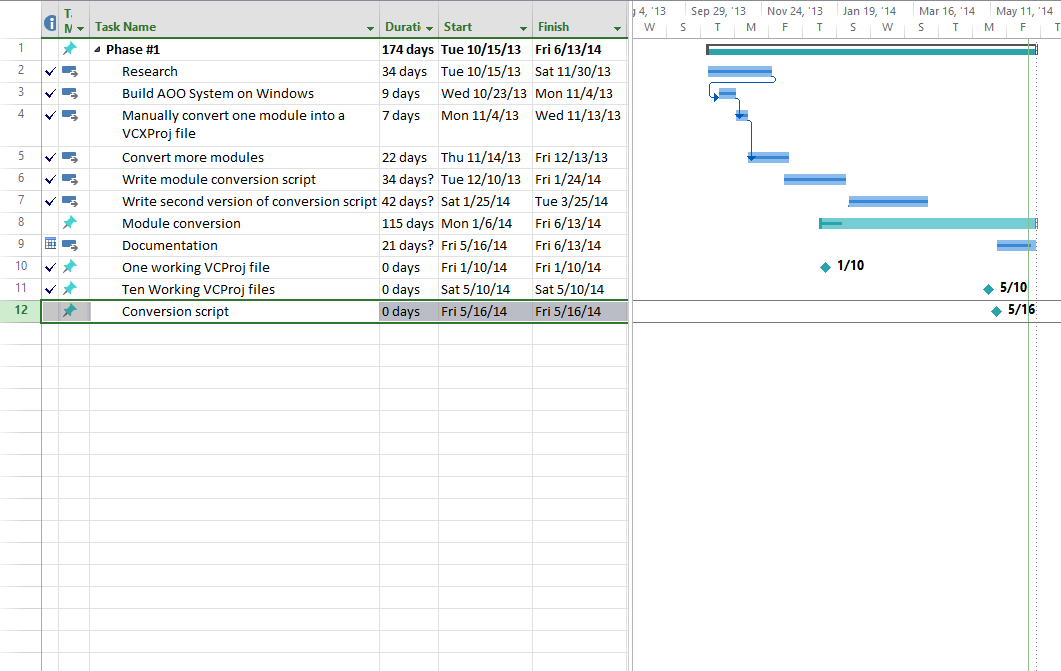
**Glossary:**

1. vcxproj: similar to a makefile in Unix/Linux systems. Determines how the project is compiled. Used to be vcproj files until more recent versions of Visual Studio. vcxproj files are written in XML.
2. AOO: Apache Open Office
3. Xerces: An XML parser developed by Apache.
4. CygWin: an emulation of Linux tools in Microsoft Windows.

**3. How did the project change since the original Client Requirements Document?**

|  |  |  |  |
| --- | --- | --- | --- |
| Req. # | Requirement | Status | Comments |
| 1 | Successfully set up Apache Open Office in Windows using CYGWin. | Complete |  |
| 2 | Convert modules to native Visual Studio projects. | In Progress | One module is done, several more are close, many in progress |
| 3 | Compose a Visual Studio solution file for all converted modules. | Pending |  |
| 4 | Write up documentation on how the new build system operates | In Progress |  |
| 5 | Write automated script to create a file with a module’s dependencies | Complete | Dependencies and libraries are gotten - build rules are not, seemingly must be done manually |
| 6 | Write automated script to convert module into a VS project | In Progress | At this stage certain parts may need to be done manually. Also needs modification to work on a single module. |
| 7 | V0.1 - convert 1 module | Complete | One module built & running within old build system - good example for others. |
| 8 | V0.2 - convert 10 modules | Complete | 10 modules completed, including a Java module |
| 9 | V0.3 - convert 35 modules | Pending |  |
| 10 | V0.4 - convert 60 modules | Pending |  |
| 11 | V0.5 - convert 85 modules | Pending |  |
| 12 | V0.6 - convert 110 modules | Pending |  |
| 13 | V0.7 - convert 135 modules | Pending |  |
| 14 | V0.8 - convert 160 modules | Pending |  |
| 15 | V1.0 - convert all modules | Pending |  |

**Updated Gnatt Chart:**



**4. Your team weekly reports (P3s)**

**Week 04**

### Progress

**Met with client, arranged another meeting with client, working on creating a calendar for group work. Have done preliminary research and downloaded relevant software and information. Started work on requirements document.**

### Problems

**None of us are well-versed in build systems. system we're looking at is difficult to understand. Requirements document doesn't map well to our project.**

### Plans

**Conduct lots of research. at least one meeting with client, another meeting with group. Finish requirements document, let client review it, finish it again.**

**Week 05**

### Progress

**We're nearing completion on the requirements document, with only a few more modifications and revisions to make. We have gotten a much better idea now of what we will need to do.**

### Problems

**We still need to get a good idea of how the build system is set up, and we haven't successfully built it yet.**

### Plans

**First priority is completing the CRD by Tuesday. Other than that, we need to get the current build system building, and then need to get to work on converting one module's makefile into a vcproj file.**

**Week 06**

### Progress

**We've put in work on the technology document and have sorted out some of the issues we had in running the VMs we were given to build the system.**

### Problems

**We still haven't gotten the system to build. We are also still conducting research into how we will structure the conversions into vcproj files.**

### Plans

**We hope to have the system building by next week, and will begin work on converting a module shortly after that. We will also have the technology document done by Thursday. In addition, we will put the technology document & CRD up on the wiki soon.**

**Week 07**

### Progress

**We spent some time together trying to work through our issues with the current build system, and are hopefully just a few steps away from finally building it. We also did some research on different ways to build the project within Windows for the Technology document, although ultimately it didn't change our preferred methodology.**

### Problems

**We still have a few problems remaining with building through the current system. Evidently Cygwin64 is incompatible, so we will be trying again with Cygwin32. We've also had numerous pathing problems with the configuration step of the build.**

### Plans

**We need to have the system building within the next week - we're behind schedule on that. After that we will put in work on converting our first module. We will also upload the CRD & Technology document to the wiki.**

**Week 08**

### Progress

**We have nearly built the project. We upped our documents onto the wiki, and have updated the virtual machine we are using so we can all build in the same environment.**

### Problems

**There are still a few build problems to sort out. We're behind schedule now, and will have to refactor our schedule going forward.**

### Plans

**Build the project, convert a module, work on project poster, reorganize schedule.**

**Week 10**

### Progress

**We now all have a VM which successfully builds the project. We've begun work documenting every module's dependencies, makefiles, and build rules, and have a clear idea of what we need to do over the Winter break.**

### Problems

**We may need to learn more about how build rules work, the syntax present in the build.lst files, and the syntax in makefiles.**

### Plans

**Over the break we will write up a document/set of documents which breaks down the information relevant to the build system for every module in the system. This will greatly help our conversion process in the future, and will serve as good documentation as well.**

**Week 11**

### Progress

**Began Project Importance document. Created script to compile requested information from modules into output files during the break.**

### Problems

**Need to sync up with client and mentor to review information contained within the text files.**

### Plans

**Sync up, make any necessary changes to the data that we gathered, and begin planning how to tackle converting a module.**

**Week 12**

### Progress

**We met with our client, and got some good direction towards what we need to do next. We discussed the work that we accomplished over the break, and talked about how we'll meet over the course of the coming term.**

### Problems

**We need to experiment with putting some modules together with the information we've compiled, and we need to figure out how build rules work in Visual Studio after that.**

### Plans

**We need to finish our block diagram by Tuesday. We are going to meet on Monday to do some work on module conversions and get some experience putting our solution together.**

**Week 13**

### Progress

**Successfully converted and tested one module.**

### Problems

**Need to devise methods of automating module creation using scripts.**

### Plans

**Continue to manually convert modules in order to uncover more issues that will need to be addressed in the script. Write a script capable of generating at least the Vcproj file for this module.**

**Week 14**

### Progress

**We have almost completed the script that makes a vcproj out of the module information we compiled. We're also working on figuring out how projects with multiple makefiles will work with Visual Studio.**

### Problems

**None at the moment.**

### Plans

**Finish the vcproj script, build as many projects with that version as possible, see where it breaks, and improve it. Incorporate build rule handling & multiple makefile handling into the script.**

**Week 15**

### Progress

**Script has progressed, pathing is improved, library files are included. Jan has a copy of the script and is going to review it.**

### Problems

**Need to consider how to move what's in the postbuild to the vcprojects for efficiency purposes.**

### Plans

**Further script work, integrating multiple makefiles, build rule gathering. Need to get script generating files that build at least some modules. Need to look at when different modules rebuild and whether this affects what postbuild does.**

**Week 17**

**Progress**

**Script has progressed, pathing is improved, library files are included. Jan has a copy of the script and is going to review it.**

**Problems**

**Need to consider how to move what's in the postbuild to the vcprojects for efficiency purposes.**

**Plans**

**Further script work, integrating multiple makefiles, build rule gathering. Need to get script generating files that build at least some modules. Need to look at when different modules rebuild and whether this affects what postbuild does.**

**Week 18**

### Progress

**Met with client about issues compiling manually converted modules, script progressed, confirmed folder structure of vcproj's, got instructions for a new script finding & replacing varnames in the vcproj's.**

### Problems

**Module compilation problems, missing files (got a solution, need to verify it fixes the problem), poster.**

### Plans

**Finish poster, work on new script & old script functionality, continue converting modules manually. Upload script to Github (finally).**

**Week 19**

### Progress

**Progress manually converting more modules and figuring out what sorts of tweaks need to be made to build specific modules.**

### Problems

**None at the moment.**

### Plans

**Upload script to Github to allow us all to work on it effectively, build modules using script, further script, further manual conversion.**

**Week 20**

### Progress

**More script work, manual conversion work. Not much done through course of dead week, I don't think.**

### Problems

**None at the moment, just other coursework taking precedence.**

### Plans

**Over the course of the break I'll have lots of time to develop the script more. We will remain in touch about any progress we make and come back next term with a little more to work with.**

**Week 21**

### Progress

**Made progress on general process of manually converting modules, now significantly closer to building several more and have a better idea of how to do it quickly & easily. Met up with group again, started establishing a new schedule for this term.**

### Problems

**The vcproj script needs some work to build a specific module, & still has a little ways to go otherwise. Need a script to replace macros with what they are mapped to. Various problems building specific modules, which we frequently contact our client about.**

### Plans

**Work on both aforementioned scripts, and work on manually converting modules, both with and without the script.**

**Week 22**

### Progress

**I (Nick) have been working intently on getting the automation up and running. Initial impressions are pretty good; its been able to convert the majority of several modules by utilizing the output of the original build system for that module. Matthew and Colin are working on manual conversion of modules in the meantime.**

### Problems

**I've been looking into setting environment variables in Visual Studio, but it doesn't seem to be possible to do so for the build portion, only for debugging the program. If you have any insight into this, that would be great. The module Icc performs file extraction using tar during the initial setup phase, and to the best of my knowledge there is no standard archive unpacker on Windows. As far as I can tell, we either need to assume the user has installed an archive program and use it, or simply have the vcproj files work under the assumption that the necessary files have already been extracted via the existing build system. Again, any insight you have would be great.**

### Plans

**I'm trying to put the finishing touches on automation before meeting during class time tomorrow in order to see if it will help Matthew and Colin.**

**Week 23**

### Progress

**Nick made great progress on the script, which now creates vcproj files which can come close to building many more modules than before. We also have a couple more modules built using this script.**

### Problems

**The script is in progress, and still has a few kinks to work out. There are also still things which may need to be tweaked for individual modules, and we're not sure yet what to do about pre & post-build events, or environment variables.**

### Plans

**Convert more modules & work through issues they have building, progress the script.**

**Week 24**

### Progress

**More script development to handle different-formatted input files, & more module conversion.**

### Problems

**Handling different formats in input files, various problems that arise with converting individual modules. Meant to meet with client last week, but didn't manage to.**

### Plans

**Convert more modules, get ourselves ready for the expo. Also need to get back in contact with the client.**

**Week 25**

### Progress

**As before, module conversion & script work. As of now, there are 7 completed modules, & several more in the works.**

### Problems

**Need to come up with a way to run Linux-based programs through Visual Studio, probably by running Cygwin during the build. Also need to work on an XML stylesheet to set environment variables for the build.**

### Plans

**The expo is coming up, and we need to try to get as many modules built before then as possible.**

**Week 26**

### Progress

**Module conversion is in progress. Preparation for expo the coming Friday.**

### Problems

**Same as before.**

### Plans

**Module conversion & documentation of the process we've underwent, & getting ready for the expo.**

**Week 28**

### Progress

**Some work on module conversion. Met with client about what's left to do. Update to the conversion script.**

### Problems

**Time is running out, need to get modules out & document.**

### Plans

**Module conversion, work on final portfolio, documenting.**

**Week 29**

### Progress

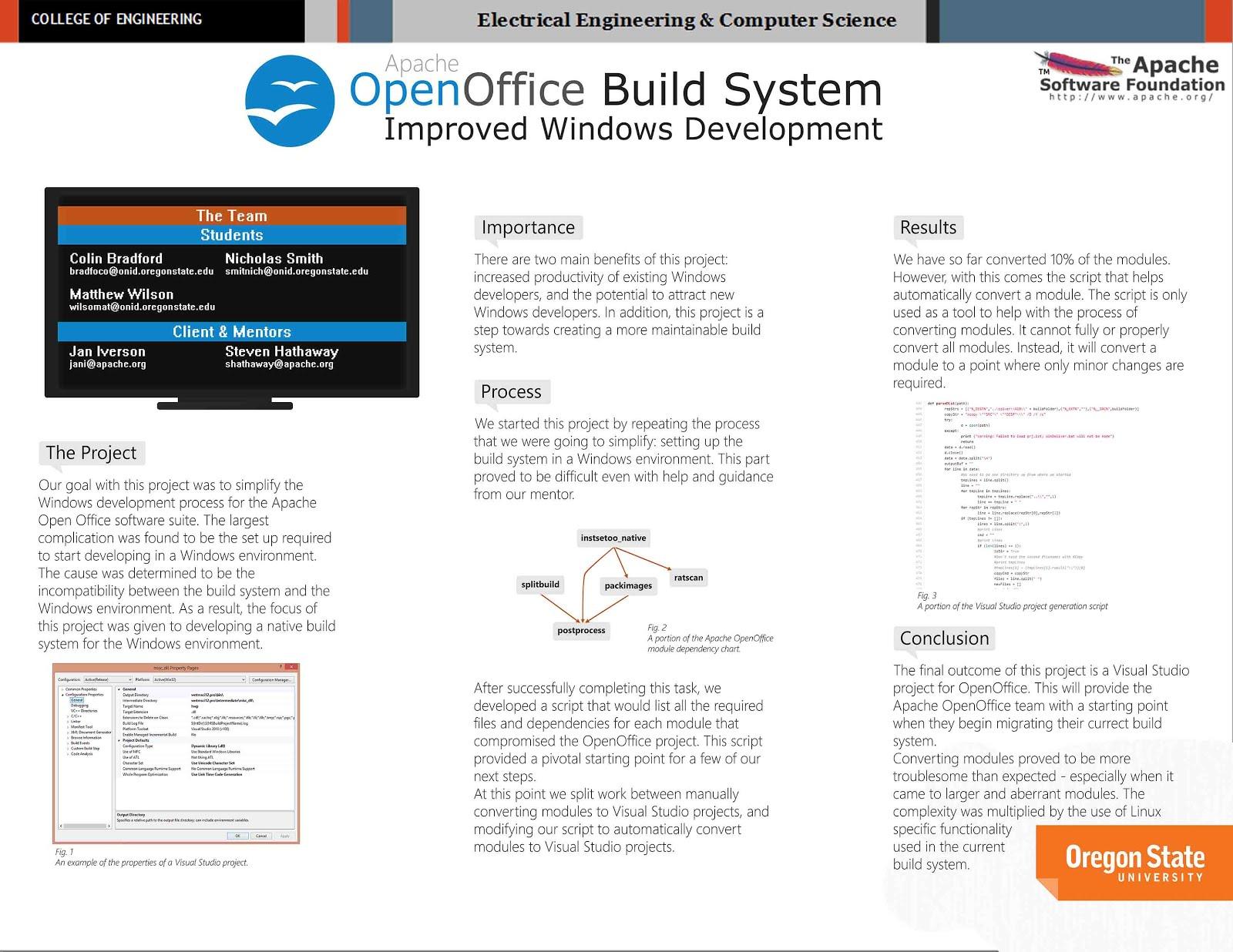
**Work on our final portfolio & wrapping up all that we've done has begun.**

### Problems

**The process underwent while converting modules is not well- documented right now. We need to disclose as much as we can about how to go about utilizing the script & working with converting modules.**

### Plans

**Work on the final portfolio, put up as much information as we can about the project on the AOO wiki, & communicate with eachother about where the project is at. Make sure the project is in a good final state by the end of the term.**



**6. Project Documentation**

**How it works**

The script works by parsing the output of how the current build system builds modules, and using relevant information to create a visual studio project file for the module. It is not meant to generate a 100% working visual studio project. The script should be able to generate a project file for all modules, and each project should have most of what it requires already. It would take far too long to manually convert all the modules, and it would also take much too long to create a script that could convert all modules perfectly. We decided that the script should be able to create a project for each module, but it should leave most projects with some amount of customization required. It reduces the amount of time spent on manually converting modules, while not requiring too much time spent writing the script.

How to use

1. Choose a module to convert.
2. Enter module directory and delete the **wntmsci12.pro** directory.
3. Enter the main directory (contains all modules).
4. Enter the **instsetoo\_native** directory.
5. Execute and pipe the output to text file created in step 4:
6. build --all:[moduleName] > ../[moduleName]/[moduleName].txt
7. Wait until the module has been built:
   1. Open the text file created in step 4.
   2. Wait until the second occurrence of **Building module**.
   3. Stop the process started in step 6.
   4. Delete everything after the second occurrence of **Building module**
8. Enter the main directory.
9. Run the vcGen.py script (format: python vcGen.py [moduleName]).

The actual deliverables are the created vcProj files. These mostly work by simply sliding into place in the existing build system. Installing the software requires first installing the existing build system and running the configure scripts. Once it is set up, the vcProj files simply need to be copied to the proper directories. There are no specific requirements to run the software other than the original requirements for building OpenOffice on Windows.

**7. How did you learn new technology?**

* What websites were helpful? (Listed in order of helpfulness.)
  + Wiki for project: http://wiki.openoffice.org/wiki/Build\_System\_Analysis:capstone2013\_windows\_build
  + Build System Analysis: http://wiki.openoffice.org/wiki/Build\_System\_Analysis
  + Apache Xerces website: http://xerces.apache.org/
  + W3 XML tutorials: http://www.w3schools.com/xml/default.asp
  + Visual Studio changes: http://msdn.microsoft.com/en-us/library/ee862524.aspx
* What, if any, reference books really helped?

We mostly used either online resources or help from our local mentor.

* Were there any people on campus that were really helpful?

We did not find anyone on campus that was particularly helpful. However, our client did have a very helpful mentor from Salem that would visit every week.

**8. What did you learn?**

**Colin Bradford**

* What technical information did you learn?

I learned how Visual Studio project files are formatted, and how they work with Visual Studio. In addition, I learned about build systems in general.

* What non-technical information did you learn?

Not much that isn’t covered by what I’ve learned about working on a project and in a group.

* What have you learned about project work?

Ideas change, and as a result some work will be obsolete. Efficient work depends on good planning.

* What have you learned about project management?

It can be difficult to plan and manage a project when there is little knowledge or understanding of the task. A plan is likely to change multiple times as your understanding grows.

* What have you learned about working in teams?

It is difficult to maintain a good level of communication between teammates. It is especially hard when meeting with clients over video call.

* **If you could do it all over, what would you do differently?**

I would spend more time during the initial planning phase researching the technologies involved. Specifically, I would spend more time understanding how Visual Studio project files work, and how the current build system works. With a better understanding of these two things, I think we could have made a much better plan.

**Nicholas Smith**

* What technical information did you learn?

Learned how Microsoft Visual Studio project files work as well as general

build system and Cygwin usage.

* What non-technical information did you learn?

Nothing in particular

* What have you learned about project work?

Juggling working on the project with general college requirements is difficult

* What have you learned about project management?

How difficult the initial estimate of time taken can be when you’re dealing

with an unfamiliar project

* What have you learned about working in teams?

Coordinating times to meet can be extremely difficult

* **If you could do it all over, what would you do differently?**

Get a better idea of how the existing build system works by carefully examining its output towards the beginning of the project, rather than later

**Matthew Wilson**

* What technical information did you learn?  
  Learned a lot about build systems, including dependencies, build rules, environment variables, & the workings of makefiles & vcproj files.
* What non-technical information did you learn?  
  Learned how to plan a project over such a long time.
* What have you learned about project work?  
  Easy to push things back without hard deadlines in mind, must have a plan moving forward, things are prone to change.
* What have you learned about project management?  
  The initial planning phase is difficult given little background. Things are very subject to change as you learn more about the project.
* What have you learned about working in teams?  
  It can be difficult to spread the workload & make sure everyone is on the same page.
* **If you could do it all over, what would you do differently?**Realize how to use the old build system to more efficiently convert to the new on. We spent too long trying to build from scratch instead of doing this. I also think that group work sessions, rather than individual contributions, would have been more effective.

**9. Appendix 1: Code**

**General Module Layout**

Since the purpose of this project was to create Microsoft Visual Studio solutions, this section will focus on the general structure of a module.

**‘module’\_cygwin:**

Contains commands to be executed using Cygwin, called as part of the \*\_prebuild VcProj file. Although we are able to move many portions of the build system to Windows, many modules still contain calls to Unix utilities such as cat, sed, etc which do not have a simple windows alternative.

**Generic VcProj Files:**

There are one of these for every target: (dll, lib or exe). Multiple targets may be made in the process of creating the final output file of the entire module. Both the compilation commands, and calls to link.exe or lib.exe are gathered from the existing build system. Dependencies must be calculated as well.

**winenv.set.sh:**

Generated by the original build system, contains settings on where files are found. Called before running any Cygwin scripts

**AOOWinGlobal**

A property sheet included with every module. Currently not used, but can be used to set any global values between modules.

**WinDeliver.bat**

Contains commands generated using the d.lst file, which contains the list of files to deliver. xcopy is used so that the files will not be copied unless they are newer than the existing files.

**versionReplace.py**

A basic Python script used to perform one of the more common shell commands (patching a file) without using Cygwin. \*\_versions.bat calls this file.

**vcGen.py**

The automation used to create most of the modules. Reads in the output from the existing build system and organizes it into a Visual Studio Solution.

**FixPaths:**

This command takes in a string containing absolute paths, or paths that are relative to a directory other than the base one, and normalizes them. This is done by replacing the known absolute path with a relative path to the main folder, as well as using the fact that the wntmisc12.pro folder is always in the root folder of a module to change relative paths to work from the root folder.

def fixPaths(cmd):

newCmds = cmd.split()

retCmd = ""

repCmd = ""

for newCmd in newCmds:

build\_it = newCmd.find(buildFolder+"/")

if (build\_it == -1):

continue

repCmd = newCmd[:build\_it]

if (repCmd.find(".") == -1):

return cmd

repCmd = repCmd[repCmd.find("."):]

#print "RepCmd found: " + repCmd

break

for newCmd in newCmds:

if (newCmd == "cl.exe" or newCmd == "link" or newCmd == "lib"):

continue

newerCmd = newCmd

newerCmd = newerCmd.replace(repCmd,"")

#if newerCmd.find("../") != -1:

if newerCmd[0] != '-' and newerCmd[0] != '/' and newerCmd[0] != '@':

#If we have a lib file that is just a filename then it must

#be a library included from outside the module

if (newerCmd[-4:] == ".lib" and newerCmd.find("/") == -1):

continue

if (repCmd == ""):

print "No wntmsci12.pro path found”

else:

foundFiles = tryToFindFile(newerCmd,repCmd,newCmd)

if len(foundFiles) > 0:

newerCmd = ""

for f in foundFiles:

newerCmd += "-I" + f + " "

for mainPath in mainPaths:

newerCmd = newerCmd.replace(mainPath + "/" + moduleName + "/","")

newerCmd = newerCmd.replace(mainPath,"..")

#print newerCmd

retCmd += newerCmd + " "

return retCmd

**FindNext:**

Iterates through the output files and finds calls to ‘cl.exe’, ‘link’, and ‘lib’ These correspond to compiling object files, making an exe or dll file, or a library, respectively.

def findNext(str,it):

findStrs = ["cl.exe ","\nlib ","\nlink "]

ret\_it = len(str)+1

fStr = ""

for findStr in findStrs:

tmp\_it = str.find(findStr,it)

if (tmp\_it < ret\_it and tmp\_it != -1):

ret\_it = tmp\_it

fStr = findStr

fStr = fStr.strip()

return ret\_it, fStr

**PatchVcProjMake:**

Takes an input Makefile vcProject, and patches certain portions of it, denoted by being surrounded with ^, with the commands to be executed. The same basic process is used to make lib and exe files as well.

def patchVCProjMake(prjName,isPostBuild,buildCmd,rebuildCmd,cleanCmd):

global preBuildId

rootFile = "make\_proj\_master.vcxproj"

f = open(rootFile,"r")

guidStr = "^GUID^"

prjNameStr = "^PRJ\_NAME^"

buildTypeStr = "^BUILD\_TYPE^"

buildCmdStr = "^BUILD\_CMD^"

rebuildCmdStr = "^REBUILD\_CMD^"

cleanCmdStr = "^CLEAN\_CMD^"

origFile = f.read()

moduleFileName = ""

if isPostBuild:

buildType = "deliver"

else:

buildType = "prebuild"

f.close()

if (isPostBuild):

moduleFileName = prjName+"/"+prjName+"\_deliver.vcxproj"

f = open(moduleFileName,"w")

else:

moduleFileName = prjName+"/"+prjName+"\_prebuild.vcxproj"

f = open(moduleFileName,"w")

prjId = ProjectGUID()

if (isPostBuild):

newName = prjName + "\_deliver"

else:

newName = prjName + "\_prebuild"

origFile = origFile.replace(guidStr,sanitizeArg(prjId))

origFile = origFile.replace(prjNameStr,sanitizeArg(newName.strip()))

origFile = origFile.replace(buildTypeStr,sanitizeArg(buildType))

origFile = origFile.replace(buildCmdStr,buildCmd)

origFile = origFile.replace(rebuildCmdStr,rebuildCmd)

origFile = origFile.replace(cleanCmdStr,cleanCmd)

f.write(origFile)

f.close()

dependencies = []

if (isPostBuild):

for prj in allProjects:

dependencies.append(prj[1])

else:

preBuildId = prjId

prjTuple = newName.strip(), prjId, "", dependencies, moduleFileName[moduleFileName.rfind("/")+1:], []

allProjects.append(prjTuple)

**10. Appendix 2: Photos, etc.**