High Performance XML/XSLT Transformation Server

Technology Review

Zixun Lu (luzi), Shuai Peng (pengs), Elijah Voigt (voigte)

CS 461 | CS Senior Capstone

Fall 2016

February 16, 2017

Abstract

This document outlines the technologies investigated and chosen for the execution of the XZES40 Transformer XML document transformer project. These technologies range from coding frameworks, to langauges, to conceptual choices made to attack each problem. Some of our requirements were imposed by our client; even in those cases we had a notable room for creativeity and choices to make.

1 INTRODUCTION

The following tables illustrates who is responsible for each component of the application.

Required functionality:

Section	Author
Research and Benchmarking	Zixun Lu
XML/XSLT Document Transformation	Shuai Peng
XML/XSLT Document Parsing	Elijah C. Voigt
XML/XSLT Document Caching	Shuai Peng
XML/XSLT Document Parallel computation	Zixun Lu
Web API	Elijah C. Voigt
Web Interface	Shuai Peng

Stretch goal functionality:

Linux Package	Zixun Lu
Command Line Interface	Elijah C. Voigt
Windows Packae	Shuai Peng
BSD Package	Elijah C. Voigt

CONTENTS

REFERENCES

[1] *Apache Xalan*. http://xml.apache.org/xalan-c/commandline.html: Apache Software Foundation.

[2] *bootstrap 3 Introduction*. http://www.w3schools.com/bootstrap/default.asp: W3schools.

[3] *Building Packages with Poudriere*. https://www.freebsd.org/doc/handbook/ports-poudriere.html: FreeBSD Documentation.

[4] *CSS Introduction*. http://www.w3schools.com/css/css_intro.asp: W3schools.

[5] *EMCO MSI Package Builder - Overview*. http://emcosoftware.com/msi-package-builder: Emco software.

[6] *Events vs. Trees*. http://sax.sourceforge.net/event.html: SAX Project.

[7] *Features Advanced Installer, publisher=caphyon*. http://www.advancedinstaller.com/features.html.

[8] *FPM on Github*. https://github.com/jordansissel/fpm: Jordan Sissel.

[9] *Getting Started With Foundation*. http://foundation.zurb.com/develop/getting-started.html: Zurb.

[10] *How to package for debian*. https://wiki.debian.org/HowToPackageForDebian: Linux Foundation.

[11] *Kore.io*. https://kore.io/: Joris Vink.

[12] *Message Passing Interface*. https://computing.llnl.gov/tutorials/mpi/: Lawrence Livermore National Laboratory.

[13] *openmp compilers*. http://www.openmp.org/resources/openmp-compilers/: OpenMP organization.

[14] *pkg-create man page*. https://www.freebsd.org/cgi/man.cgi?pkg-create%288%29: FreeBSD System Manager's Manual.

[15] *pthread libraies*. http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html: YoLinux Tutorial.

[16] *RaptorXML Server*. https://www.altova.com/raptorxml.html: Altova Foundation.

[17] *Sablotron*. http://freecode.com/projects/sablotron: free code.

[18] *Saxon/C*. http://www.saxonica.com/saxon-c/index.xml: Saxonica.

[19] *Set up RPM environment*. https://wiki.centos.org/HowTos/SetupRpmBuildEnvironment: Linux Foundation.

[20] *WiX Toolset Tutorial*. https://www.firegiant.com/wix/tutorial/: Firegiant.

2 RESEARCH AND BENCHMARKING

2.1 Options

2.2 Goals for use in design

Our team will do the research and benchmarking to grantee that our application is fast enough.

2.3 Criteria being evaluated

We will test some number of requests against a comparable to find which transformers use the time less. Throughout development we will put our application through the same paces and compare which is faster.

2.4 Comparison breakdown

Technology	Description
Xalan CLI [1]	• Xalan-C++ uses Xerces-C++ to parse XML documents and XSL stylesheets. • The project provides an open source CLI program to test the project libraries. • Free and Open Source
Altova [16]	• To meet industry demands for an ultra-fast processor. • It offers powerful, flexible options for developers including cml, python. • Superior error reporting capabilities include reporting of multiple errors, detailed error descriptions.

2.5 Discussion

Xerces is a simple CLI application developed by the Xerces project to test the library. This is very similar to our program as it is open source, uses the same libraries, but lacks the caching we will implement.

RaptorXML is built from the ground up to be optimized for the latest standard and parallel computing environments. It is proprietary tool which we may try to out-perform as a stretch goal, but to start with out application will not try to out-perform.

2.6 Selection

We will compare our application to the Xalan-C CLI as it is the closest competitor to our application.

3 XML/XSLT Document Transformation

3.1 Option

There are three options for the XML/XSLT Document Transformation. The first option is Xalan-C++, and the second option is Saxon C, the third option is Sablotron.

3.2 Goals for use in design

The major function of XZES40-Transformer is XML/XSLT document transforming. Apache foundation provide many ways to achieve this function.

3.3 Criteria being evaluated

The XML/XSLT document transformation is our major function for XZES40-Transformer application. We want this transformation compatible with good parse tools, and version of XSLT and XPath.

3.4 Comparison breakdown

The first option is Xalan-C++, this technology is required by our client, and this technology is supported by Apache. Xalan-C++ is an XSLT processor for transforming XML documents into HTML, text, or other XML document types. Xalan-C++ contain the XercesC tools, so we don't need consider the parse tools.

The second option is Saxon C, which is Saxon company project. It performs the same operations as Xalan-C++, but it supports different version of XSLT and Xpath.

The third option technology is Sablotron. It used same version of XSLT and Xpath with Xalan-C++, but it designed to be as small program.

Technology	Description
Xalan-C++ [**xalan**]	• Xalan-C++ is open source project developed by Apache. It is implemented by XSLT version 1.0 and XPath version 1.0. • Xalan-C++ uses Xerces-C++ to parse XML documents and XSL style sheets.
Saxon C [18]	• Saxon C is open source project. It is implemented by XSLT 2.0/3.0 version and XPath version 2.0/3.0. • Saxon C use different parse tools to handle the date.
Sablotron [17]	• Sablotron is open source project by gingerall. It is implemented by XSLT 1.0 and Xpath 1.0. • Sablotron need extra parse tools to complete transformation.

3.5 Discussion

The Xalan-C++ is the most powerful and popular XML/XSLT document transformation library and is used in many propetary as well as open source tools. It a robust implementation of the W3C recommendations for XSL Transformations (XSLT) and the XML Path Language (XPath). The Xalan-C++ is continuing update, and it is good tools for transforming documents. However, Xalan-C++ has poor structure API reference, and it makes developer hard to read and understand. Thus, developer need spend time on doing research with API.

The Saxon C is a good alternative to Xalan-C++. Saxon C can handle higher version of XSLT and Xpath, however implements different parse tools, so we need find out other technology.

The Sablotron does the same work as the Xalan-C++, but Sablotron is designed to be as compact and portable as possible. The size of Sablotron is much small than Xalan-C. However, Sablotron is old and has not been updated it since 2006.

3.6 The Best option

We will choose Xalan-C++ as our solution technology. The first reason is that Xalan-C is required by our client. We want to using Xerces-C as our parse tools, so Xalan-C is the best choice. Xalan-C++ is easy to use with clear example.

The second reason is that Sablotron is not stable. Even they are open source project, but the community has forgetten it. Saxon C is a good second option if client require different tools to complete XML/XSLT document transformation.

4 XML/XSLT Document Parsing (Elijah C. Voigt)

XML document compilation is the process by which an XML formatted document is taken and parsed into an in-memory object. The library we will be using, as requested by or client, will be the Xerces-C/C++ XML parser library; this review will evaluate that library. The other point of wiggle-room we have is in what way we parse and store the document in memory.

4.1 Options

For this requirement we will review the C/C++ library that we were requested to use by our client, and two XML parsing techniques we may use. The two techniques for parsing an XML document include SAX and DOM, the library we need to use is able to perform both methods. These are both designed with specific use-cases in mind, however the method we ought to use is not so simply chosen.

4.2 Goals for use in design

With the goal of optimizing peformance in mind we will choose the parsing option which is fastest for our application. This is not as simple as choosing the one which is known for being fastest, because we will also be caching documents and performing parallel computations which may pull the needle toward one method of compiliation over another.

4.3 Criteria being evaluated

We will be evaluating the parsing method which is likely to perform best in our application given that we will be caching documents and compiling them in parallel.

4.4 Comparison breakdown

Technology	Description
Xerces-C/C++ [**xerces**]	• Requested to be used by our client. • Implements DOM parsing. • Implements SAX parsing. • Feature Rich.
DOM parsing [6]	• Memory intensive. • Ideal for carrying out many operations on a document.
SAX parsing [6]	• Memory light. • Event-based processing. • Ideal for minimal transformations.

4.5 Discussion

DOM parsing is know for being memory intensive and slower in simple cases. It is also know for being ideal in applications like updating a web-page where many operations happen to a document. It is not necessarily ideal for simple document transformation, but does produce a complete parsed object which we can cache easily.

SAX parsing is known for being better than DOM parsing in that it parses a document in an event based method, giving it a smaller memory footprint and shorter time-to-transform. Simple transformations are common use-cases for SAX parsing. SAX does not produce an object model and instead uses callbacks to perform document transformations.

4.6 Selection

We will be using the Xerces-C/C++ library to accomplish the task of XML document parsing. It is feature rich enough to give us leeway in our development where we need it The library is also lean enough that it should not affect performance negatively.

As for choosing between SAX parsing vs DOM parsing, we will most likely choose DOM parsing since it fits our application's caching requirement well. DOM parsing produces an easily cached tree object which we can store and retrieve for later operations. SAX parsing is faster than DOM parsing, but the time saved by using a cached object instead of re-parsing an XML/XSLT document dependency will likely outweigh the benefits of SAX parsing. It may be worth-while to investigate using SAX parsing early on in development if we find a notable performance boost, so some amount of SAX proof of concept work should be done for the sake of being thorough, but DOM parsing will be the targeted document parsing method.

5 XML/XSLT Document Caching

5.1 Option

There are three technology options for cache. The first option is storing cache into memory. The second option is storing our cache in binary file on disk. The third option is to create database to handle all of data.

5.2 Goals for use in design

Caching is the "plus one" function of XZES40-Transformer application. Other similar application compile files each time and this wastes a lot of time and resource. We will create cache to solve this problem.

5.3 Criteria being evaluated

We want save the time and resources in our XML transformer, so efficiency is the most element that we consider. This is not only the speed of reading and writing from the cache, we must also weigh the persistence of the cache to avoid recompiling when the system (application or host) is restarted.

5.4 Comparison breakdown

The first option is storing the cache in-memory, this is the faster and easy way to store cache.

The second option is to create a binary file with the cache. When we are run our application, we read the cache data in from the cache file into memory. When the cache is updated it is written back to the original file.

The third option technology is that we create database to handle memory. This spends time to design and create database.

Technology	Description
Memory	• Application check data from memory, and put cache into memory. • Retrieving data from memory is the faster way.
Temporary binary file	• Application loads binary file when it starts. After we close it, application save binary file in external storage driver. • Loading temporary binary file spend time, so it is slower than memory.
Database	• Application access data from database. • It takes time to create and manage a database.

5.5 Discussion

Storing cache into memory is the most easy way, we just need to allocate memory. However, the main drawback of this technology is when we close application, all of cache data will be wiped out. We have to compile file next time when we start running the application. When we are developing the XZES40-Transformer, we find a tools that KeyList is built in XercesC. This tools is helpful for managing memory, and it has good data structure.

Creating a binary file can avoid losing cache data, but it spends time to load file into memory when application starts.

Creating database is bad option for XZES40-Transformer application because access database spend resource, and it waste time to search cache data.

5.6 Selection

The best option technology is that storing cache into memory. Although it will lost data after close application, it save the time, and it the faster way. We may add a 'backup cache' solution to make this the best of both worlds, restoring from the backup when the system restarts but working mostly in-memory. And we will using KeyList, because it is built in XercesC. We can just add API, and easy to control the cache storing.

6 Parallel Document Transformation

6.1 Options

6.2 Goals for use in design

XZES40-Transformer will be increased performance by using parrallel computation. The parallel processing of the containment queries against an XML document utilizes parallel variants of the serial algorithm. There are two ways to do the prallel computation.

6.3 Criteria being evaluated

XZES40-Transformer use the parallel computation method do the same processing and operating in parallel. The criteria for a good tool in this area is mostly how easy to write code it is with the tool and how maintainable the code is, and of course how fast the eventual program is.

6.4 Comparison breakdown

Technology	Description
POSIX Threads [15]	• Defines a set of C types, functions and constants • Spawn concurrent units of processing • Achieve big speedups, as all cores of CPU are used at the same time.
OpenMP [13]	• An API that implements a multi-thread, shared memory form of parallelism. • Uses a set of compiler directives • Take care of many of low-level details
MPI [12]	• Core syntax and semantics of library • Complexity, scope and control • Manage allocation, communication, and synchronization of a set of processes

6.5 Discussion

Pthreads is a standard for prgramming with threads, it can achieve big speedups, as all cores of your CPU are used at the same time. OpenMP uses a set of compiler directives that are incorporated at compile-time to generate a multi-threaded version of your code. MPI allows us to manage allocation, communication, and synchronization of a set of process.

6.6 Selection

We will use MPI because it is a high-level standard, and so it will be easy to develop with.

7 Web API (Elijah C. Voigt)

7.1 Options

Our web API may be implemented via a C++ web application, a Python or Ruby web application, or an Apache webserver CGI script. Each of these has pros and cons, and each get the job done at some cost and with some benefits.

7.2 Goals for use in design

XZES40-Transformer will be accessible via a web API. This can be implemented a few different ways, but all of them must accomplish the same goal of allowing people to use the service over a network. The three options being evaluated here vary in how they achieve this goal, and so they represent more their technology and less the specific implementation which will be used.

7.3 Criteria being evaluated

The core of this application is related to document transformation. The more time that is spent on non-document transformation tasks should be kept to a minimum. Any technology we use to implement our web API should be simple, easy to develop, and easy to maintain. In short, *keep it simple stupid*.

7.4 Comparison breakdown

Our first option is to use an Apache CGI script, which would be a simple Python, Perl, or Ruby file which calls our C program and returns the results (transformed document or error) to the user, all via an Apache server gateway. The second options is to write a native web application using a web-app framework like Kore to handle HTTP requests. The third option is to use a python framework like Flask to handle HTTP requests. Each of these would be something that calls our document transformation app and exposes it to the outside world. How we handle that is important to consider.

Technology	Description
Apache CGI Script [**cgi-tutorial**]	• HTTP requests are handled by the Apache web-server. • XZES functionality is called by "shelling out" to the program and returning the results. • Apache CGI Script requires a running Apache Server on the host.
Kore web-app framework [11] [**kore-feature**]	• HTTP Requests are handled by the Kore framework. • XZES functionality is called natively with C code. • Kore web-app framework acts as an independent daemon.
Flask web-app framework [**flask-site**]	• HTTP Requests are handled by the Flask framework. • XZES functionality is called either natively or by using `exec` to "shell out". • Flask web-app framework can act as an independent daemon or be managed by Apache.

7.5 Discussion

The above three technologies are all entirely valid choices for our application; each approach the problem from different angles.

The Apache CGI choice is the Occam's Razor solution relative to the others. Using the Apache web server we can register a script (written in Python, Perl, Ruby, etc) to accept requests and return a response. This is simple, maintainable, and easy to create; this is especially appealing if we are only concerned with implementing the API and not fancy features like accessing a database or storing user sessions. This option also allows us to leverage existing Apache Web-server power like load balancing and simple authentication without needing to write those features ourselves, they're just an Apache configuration file away from being a reality.

Kore is a web-app framework which would allow us to develop our application in C/C++, which has it's pros and cons. C/C++ is notoriously difficult to write, and harder to write *well*, so it may be a time-sink. That said, it is nice to have a project which is written entirely in one language as contributors (ourselves and others) do not need to learn multiple languages to contribute to the transformer project.

Flask is another web-app framework, but one which is substantially easier to read and write. This has the benefit of being easier to maintain than a Kore framework, and we can write exactly the level of complexity we want from our API. On the other hand we would need to maintain a knowledge base of python, python frameworks, and python dependencies, so this is versatile but ultimately not necessarily easier to maintain than a kore framework.

7.6 Selection

Since we are working with Apache on this project, we want to develop a simple solution to our API problem, and the Apache Web server is a powerful tool we will choose to use this in our design. We will write a simple CGI script (which calls our C binary) and hook this into an Apache Web server. We will need to depend on the Apache Web server for our project's package, but this should not be as hard as writing a web app ourselves. We can also use the server to easily host our Web UI, which is a nice bonus.

8 Website UI

The Web Interface will be a simple webpage which calls our web API. For this reason this section focuses mostly on design technologies and less on HTTP request-handling technologies.

8.1 Option

This document reviews three possible technologies we can use to implement our website . First is plain-text HTML, CSS and Javascript, second is the Bootstrap front-end framework, and third is the Foundation framework.

8.2 Goals for use in design

XZES40-Transformer will have a web-based user interface, which will also be our main user interface. In considering which technology to use we will focus on making it conform with modern website design practices. This will allow us to write an application which is hopefully user friendly and intuitive to use. The website should also be look good.

8.3 Criteria being evaluated

The most important aspect to consider for this user interface is the appearance. We want to create web pages that work well on most screen sizes. So the cost, efficiency, visual appeal, and dynamic screen adjustment are what we consider most important in our technology of choice.

8.4 Comparison breakdown

- Cost: All of our options are open source and totally free and offer free documentation / tutorials.
- Efficiency: Using plain CSS/Javascript/HTML will perform well on most end-user's web-browsers, however it will be difficult to optimize the website to be responsive and adjust for smaller screen sizes. Bootstrap and Foundation were created with responsive design in mind. Both of them can change the size automatically in different size of screen. Both of them provide templates for creating web pages, but plain CSS do not provide.
- Learning speed: CSS is the basis of HTML style sheet. It is easy to understand and learn, however it hard to make it look good. Bootstrap and foundation spends time to learn, but after we learn the basically knowledge, bootstrap and foundation will be faster than CSS.

Technology	Description
CSS [4]	• CSS is open source, and it the basis style sheet for HTML. • CSS is not able to easily create a web page that fit in different size of screen. • CSS is easy to learn, but it hard to use to make a good website.
Bootstrap [2]	• Bootstrap is open source project with good forum support. • Bootstrap is efficient because it has responsive deign. It provide many templates. • Bootstrap is easy to learn and use.
Foundation [9]	• Foundation is open source project. • Foundation is efficiency, and it has responsive deign. It provide many templates. • Foundation is easy to lean and provides free tutorials.

8.5 Discussion

The table show us that all of them did similar work however, they have different advantages and drawback. CSS is basis of HTML style sheet, and it works great, but it hard to create web pages beautiful using *just* CSS and HTML. This make it hard to move web page into different size of screen, so we don't want to take CSS as our solution technology.

Bootstrap and foundation do similar work, and both are open source and free to use. However bootstrap is much more stable and provides more templates, and there is free instructions in the W3C school for using it.

8.6 Selection

The best option is the Bootstrap for our project because Bootstrap is open source project and has good tutorials in W3C schools. Bootstrap is the most popular font-end framework for web design, and it still updated by thousands of people. During we are developing our code, I think that we should have second choice. The Second choice is plain-HTML interface. Althoght plain-HTML dose not look beatiful, it is most straight way to present information to user. Keeping simple is the best way.

9 Debian & Centos Package

9.1 Options

9.2 Goals for use in design

Our team will release Debian and Centos packages. Users can directly download these from the website and use these packages to install XZES40-Transformer on their Debian and Centos operation system.

9.3 Criteria being evaluated

We may use Centos and Debian tools to build our package. We can use Centos operation system to build Centos packages and use Debian operation system to build Debian packages. Those tools are free to use.

9.4 Comparison breakdown

Technology	Description
Centos packaging tools [19]	• Use in Centos operate system • It is esay to use • Free.
FPM [8]	• Translates packages from one format to another. • Allows re-use of other system's packages. • Free.
Debian packaging tools [10]	• Use in Debian operate system • It is easy to use • Free.

9.5 Discussion

The above tools are all valuable. If we were going to just develop a Debian package we may only use the Debian tools, and the same goes for CentOS. These tools are good at creating packages for those specific platforms, but since we intend to develop tools for mutiple platforms (Linux and BSD) using FPM to create cross-OS packages would be very convenient.

9.6 Selection

In the end we will use FPM to develop our packages becuase it makes life very convenient.

10 Command Line Interface (Elijah C. Voigt)

10.1 Options

For the CLI we can develop a native C client, a simple Bash client, or split the difference with a Python client.

10.2 Goals for use in design

Each of these CLI tools must be able to construct an HTTP POST request with two documents, send that request to a server, and parse a response payload.

10.3 Criteria being evaluated

The criteria for this, as it is not part of the core functionality, is simplicity to write, maintain, and use for end-users.

10.4 Comparison breakdown

Technology	Description
C/C++ []	• Keeps the code base exclusively C/C++. • Can be distributed with system package managers. • Difficult to write and maintain.
Python []	• Simple to write and maintain. • Requires few external dependencies. • Can be distributed with the `pip` package manager.
Bash []	• Very simple to write. • Quick to write using existing system tools like `curl` or `wget`.

10.5 Discussion

The C/C++ option is not necessary at all. This is an option which ought to be considered, but ultimately isn't worth pursuing as it will be very complicated to write and maintain, especially when simpler script-based options exist.

Python is a great middle ground. The language comes with many web-request libraries and provides tools for users to upload their application to the `pip` package manager. This means we can write a tool which performs well, is easy to maintain, requires few external dependencies, and can be downloaded by a package manager. It will also be available on all platforms which run python, which includes Unix and Windows systems.

Bash is a viable candidate, especially when other tools like `curl` and `wget` will make quick work of the task. The downside to this is that we cannot host our CLI on any standard package manager for verifiable distribution. This forces users to download the CLI from our website directly. Ultimately this will probably be used as a proof of concept, but it is not a final product.

10.6 Selection

We will start by creating a CLI in Bash for testing purposes, and if time allows we will create a more polished CLI in Python. Python language offers the best of both worlds in terms of simplicity, maintainability, and ease of use for end users, but for the sake of "getting something out the door" we will first create a tool that works in Bash.

11 WINDOWS PACKAGE

11.1 Option

There is three option technology for the windows install package. The first one is WiX Toolset, the second is EMCO MSI Package Builder, and the third is Advance Installer.

11.2 Goals for use in design

XZES40-Transformer may be packaged to run on multiple platforms. We want to create install package, so users do not need to install our package manually from source.

11.3 Criteria being evaluated

Although windows install package is our option work for our project, we want it move from other platform smoothly. We want it easy to learn how to use this technology, and we also want it free. The flow of install should be simple for user, so user can just install and run application quickly.

11.4 Comparison breakdown

- Cost: WiX Toolset is open source and free software however the advanced installer is a proprietary tool which must be paid for. EMCO MSI Package Builder provide free version for individual developer, but free version of package builder limit the functionality for create MSI package.
- Security: WiX Toolset is open source project since 2004. Even the Microsoft also uses WiX to create MSI package, so WiX tools should be the most safe tools than other tools. The security of Advanced installer is acceptable. The company is created since 2002, and other big company used this tools, such as Sony, Dell, etc. EMCO MSI Package Builder is same as the advance installer.
- Stable: WiX Toolset is open source project, and there is thousand of developer contribute on this project, so WiX Toolset is more stable than other open source alternatives.
- Learning Speed: WiX Toolset has a steep learning curve. It is better to understand the fact of Microsoft install package before using this tools. Advanced installer and EMCO MSI Package Builder provides GUI for user, it is easy to using and learning.

Technology	Description
WiX Toolset [20]	• WiX Toolset is open source project • WiX Toolset is more stable and security than other tools • WiX Toolset have steed learning curve, but it is worth to learn • WiX Toolset is free for every one.
EMCO MSI Package Builder [5]	• EMCO MSI Package Builder is not open source. • EMCO MSI Package Builder is stable, and security is acceptable. • EMCO MSI Package Builder is easy to learn, because it have GUI. • EMCO MSI Package Builder has free version for individual developer.
Advance Installer [7]	• Advance Installer is close source tools. • Advance Installer is stable and security. • Advance Installer is easy to learn. It has GUI, and forum support. • Advance Installer is expensive. No free version.

11.5 Discussion

The WiX toolset is a collection of tools which build Windows installation packages from XML source code. Traditional setup tools used a programmatic, script-based approach to be installed on the target machine but WiX uses a different method. It is uses an XML configuration file to describe the steps of the installation process. Microsoft also uses WiX with all its major software packages, like Microsoft Office. [20]

EMCO MSI Package Builder is an installation tools designed for help developer create, maintain and distribute Windows Installer packages. [5] It helps developer create MSI package automatically by using changes tracking technology, which is used to generate installation project data, or manually by using the visual editor.

Advanced Installer is GUI tool that can simply create Microsoft Install packages. Advanced Installer create and maintain installation package, such as EXE, MSI, ETC. It based on the Windows Installer technology. [7]

11.6 Selection

After comparing all of above methods we have chosen to use WiX as the solution to our Windows packaging problem. The first reason why WiX is my solution technology is that it is open source software. It a powerful set of tools available to create our Windows installation package.

The second reason is that it represents by source code rather than GUI. GUI may be easy for developer, but it also hided every thing behind the GUI. If we get some weired problem, we can't solve it via GUI software.

Third reason is that it can be completely integrated into our application build processes. When install progress start, other setup modification are made in parallel, so no vital information will be lost.

The fourth reason is that it use XML as the main language. We don't need spend time on learning how to use WIX.

The setup program will complete together with the application itself. This is not required by client, but if client want to different way to create windows install package, we can choose advanced installer as our alternate technology.

12 BSD Package (Elijah C. Voigt)

12.1 Options

XZES40 Transformer will be created with an installation package so that it can easily be installed on a host system. One platform we will create a package is FreeBSD. This package will be a stretch goal. This can be created manually, with FreeBSD system tools, or with a more general system-package creation tool.

12.2 Goals for use in design

In an ideal world this tool would be easy to use, and would automatically take our source code, compile it, and package it for distribution. Unfortunately that isn't a feasible solution for such a small project, but in our design we will be considering anything that gets us close to that reality.

12.3 Criteria being evaluated

The technology chosen should be easy to use, and allow for modifications to the host package in an expedient way. Creating system packages can be a pain, but creating one *incorrectly* is even worse. The tool chosen should be free, relatively painless to use, and require minimal human interaction to prevent the introduction of human errors.

12.4 Comparison breakdown

Technology	Description
Poudriere [3]	• Creates builds for multiple platforms including other versions of FreeBSD and other CPU architectures. • Builds packages in parallel. • Free.
FPM [8]	• Translates packages from one format to another. • Allows re-use of other system's packages. • Free.
pkg-create [14]	• Simple to use. • Installed on most FreeBSD systems. • Free.

12.5 Discussion

Each of these three technologies offers similar end results, but some offer a better development experience or a more feature rich pipline.

Poudriere is appealing because it is by far the most feature rich option available. It essentially offers a system for us to test and build our software all in one tool using the BSD Jails system. The consequence is that this package would need to be created manually each time we want to release an update to our software, or just to make changes to the package.

FPM is appealing mostly because this FreeBSD package will be a stretch goal. FPM will be able to take a Debian or CentOS package *as well as* produce a FreeBSD package. This will require minimal intervention, and as long as we can test that the package produced is correct we can use this for future iterations of the package. The downside is that this is not guaranteed to work correctly as it does not provide any testing infrastructure for the package produced, so some amount of manual testing will be required.

The last option is not appealing, but viable all the same. pkg-create can be used to create a package on FreeBSD. The tool is convenient to acquire, and allows us to create the package, but does not offer nearly as many benefits as the others.

12.6 Selection

For convenience we will use FPM. Once we have a Debian package creating a FreeBSD package should be smooth sailing. The solution is free, requires minimal human interaction, and allows us to create reproducible results (i.e., weather the package works or not is not determined by the package creator's coffee intake).

13 Conclusion

To summarize our findings:

Section	Author	Technology Choice
Research and Benchmarking	Zixun Lu	?
XML/XSLT Document Transformation	Shuai Peng	?
XML/XSLT Document Parsing	Elijah C. Voigt	Xalan C++
XML/XSLT Document Caching	Shuai Peng	?
XML/XSLT Document Parallel computation	Zixun Lu	?
Web API	Elijah C. Voigt	Apache + Python CGI script
Web Interface	Shuai Peng	?
Linux Package	Zixun Lu	?
Command Line Interface	Elijah C. Voigt	Bash proof of concept, Python final product
Windows Packae	Shuai Peng	?
BSD Package	Elijah C. Voigt	FPM package creation toolkit