High Performance XML/XSLT Transformation Server

Client Requirements

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

CS 461 | CS Senior Capstone

Winter 2017

February 17, 2017

Abstract

This document outlines the client requirements for the OSU CS Capstone project "High Performance XML/XSLT Transformation Server". This covers what will be required of the application to perform well, the interfaces the application will have, optimizations the application will have, and restrictions on the application.

1 INTRODUCTION

1.1 Purpose

This document is intended to present a detailed description of the high performance XML/XSLT transformation server being developed by the Oregon State University CS Capstone Team "XZES40". The intended audience for this document are the developers and sponsors of the project.

1.2 Scope

The name of this software, for lack of a better one, will be XZES40-Transformer.

The core product being delivered is a high performance XML/XSLT transformation server. This server will be able to perform repetitive document transformations quickly and efficiently by caching previously processed and compiled documents. Time is saved by pulling from an in-memory cache of documents and their compiled state rather than downloading and compiling documents which have already been processed, as current systems tend to do. XZES40-Transformer will also carry out transformations in parallel. It will transfer documents to and from clients via the HTTP or HTTPS protocol.

The target platform for XZES40-Transformer will be Debian Linux 8 ("Jessie"). The core product will be designed to allow the program to be ported easily from Linux to other operating systems like Windows and BSD.

In addition to the core server there will also be a command-line interface and web-interface developed to interact with the application, these will be called XZES-CLI and XZES-Web respectively.

1.3 Definitions, acronyms, and abbreviations

Below is a list of acronyms and abbreviations used throughout the document:

- Extensible Markup Language XML: The human-readable data format used and processed by our application.
- Extensible Stylesheet Markup Language (XSLT): The human-readable format used to transform documents in our application.
- Xerces-C [**xerces**]: One library used to perform XML transformation in C.
- Xalan-C [**xalan**]: One library used to XML transformation in C.
- ICU [**icu**]: One library used to process UTF-8 character formated documents.
- Hypertext Transfer Procol (HTTP/HTTPS): The protocol over which XZES40-Transformer will interact with remote clients.
- HTTP Application Programming Interface (HTTP API): A standard way of communicating with a web application.
- Unified Resource Locator / Identifier (URL/URI): Addressable location of a resource over the internet (e.g., a website address).
- Debian 8 ("Jessie"): The target Linux-based operating system XZES40-Transformer will run on.
- Unicode Transmission Format 8 (UTF-8): The international standard for encoding text-based data.
- Apache Web-server: A Free and Open Source web-server.
- Common Gateway Interface Script (CGI Script): Server-side scripts that can run applications on client's behalf.

1.4 References

REFERENCES

[1] *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. https://www.w3.org/TR/2008/REC-xml-20081126/: W3C Organization.

[2] *XSL Transformations (XSLT) Version 1.0*. https://www.w3.org/TR/1999/REC-xslt-19991116: W3C Organization.

CONTENTS

2 OVERALL DESCRIPTION

The following sections of this document outline the factors that affect the creation of XZES40-Transformer at a high-level.

2.1 Product perspective

2.1.1 System interfaces

The XZES40-Transformer application will interface with the outside world over the internet via the HTTP networking protocol. The application will recieve HTTP POST requests to the application URI endpoint containing the documents to be transformed. Once the transformation is completed the transformed document will be sent to the user for download.

In addition to the transformed file the application will respond with an HTTP **OK** status. If an error occurs it will respond with a **SERVER ERROR** status and no file.

2.1.2 User interfaces

XZES40 will have two user interfaces which will access it's functionality over the internet.

- A Website to access XZES40-Transformer via a web-browser. This interface will be called "XZES40-Web".
- A CLI to access XZES40-Transformer via a terminal interface. This interface will be called "XZES40-CLI".

Both interfaces will not perform local document transformation. They will instead access the transformation service over the internet, making the HTTP API convenient to use.

2.1.3 Hardware interfaces

XZES40-Transformer will not have any direct physical interfaces as it is meant to be interacted with over HTTP or HTTPS. Any computer with an internet connection, monitor, input methods, and web-browser will be able to access XZES40-Transformer via the web interface. Any computer with an internet connection, monitor, input methods, and which has the CLI installed will be able to access XZES40-Transformer via the CLI interface.

The application will be targeted to run on a Debian Linux 8 ("Jessie") X86_64 CPU architecture server. This machine should have one port open for communicating over HTTP (port 80) and one for HTTPS (port 443) if that is configured.

2.1.4 Software interfaces

Below is a list of software required for the XZES-40 (on the host and on remote systems).

Name: Xerces-C++ XML Parser

Mnemonic: Xerces-C

Specification Number: XML 1.0 specification is implemented.

Version Number: 3.1.4 (Recent)

Source: http://xerces.apache.org/xerces-c/

Name: Xalan-C++ XSLT Processor

Mnemonic: Xalan-C

Specification Number: XML 1.0 specification is implemented.

Version Number: 1.10 (Recent)

Source: http://xalan.apache.org/xalan-c/

Name: International Components for Unicode

Mnemonic: ICU

Specification Number: Unicode 9.0

Version Number: ICU 58

Source: http://site.icu-project.org/download/58

Name: Apache CGI Processing

Mnemonic: Apache CGI

Version Number: Apache 2

Source: http://apache.org

2.1.5 Communications interfaces

An internet connection between the host and client is required for use of XZES40-Transformer. The host and client will be communicating over HTTP or HTTPS through the web interface or CLI interface. The application may be deployed behind a firewall.

Administrators may deploy multiple instance of the application, however additional instances will not be designed to communicate together, so they will each act autonomously.

2.1.6 Memory constraints

XZES40-Transformer will depend heavily on an internal caching system, so an appropriate amount of memory should be dedicated to the application. The specific amount of memory will depend on how much an instance of the application is expected to be used, however a minimum of 4GiB should be dedicated to the machine it is running on. That said, the more the merrier.

2.1.7 Operations

XZES40-Transformer will initially target the Debian Jessie operating system. To install the application a system administrator will acquire a Debian installation package (`xzes40-transformer.deb`), run the installation file, and begin the newly installed service.

The steps will roughly be as follows:

```
Download xzes40-transformer.deb
$ wget http://example.com/xzes40-transformer.deb

Install the package
$ dpkg -i xzes40-transformer.deb

Enable the xzes40-transformer Systemd service
$ systemctl enable xzes40-transformer

Start the Systemd service
$ systemctl start xzes40-transformer
```

Installation on an additional Debian system would require the same procedure as listed above.

Installation on a non-Debian operating system will require a system-specific installation file, which we will create. The non-Debian systems we will include:

- Windows 7+
- MacOS
- FreeBSD
- RedHat Enterprise Linux

Once installed and setup, XZES40-Transformer will require minimal user-interaction by the system administrator. The application will run as a daemon on the host system. If a fatal error occurs the daemon will restart.

While users are not interacting with the application it will idle in the background. During periods of intense use the application will manage it's own resources to avoid breaking.

As the application does not store ephemeral data, there will not be a need for data backup nor data restoration.

2.1.8 Site adaptation requirements

XZES40-Transformer will use Apache to manage web-requests. If users want to setup secure communication over HTTPS they will need to do this manually using Apache.

The application will include a configuration file to specify resource limits, and other relevant information. This file should be tailored to a given user's installation and needs.

In addition to the configuration file, the user may configure the daemon through the daemon manager (`systemd` for instance) to set hard-limits on the application's resource usage.

2.2 Product functions

XZES40-Transformer will perform one function: XML/XSLT document transformation. Given one XML and one XSLT documents it will return a transformed XML document.

This functionality will be remotely accessible via an HTTP web-page and CLI interface.

2.3 User characteristics

The **user** of our application is expected to have common web-interface knowledge (e.g., they should know how to navigate a website, upload a file, and download a file).

2.4 Constraints

XZES40-Transformer will be subject to the following limitations:

- The code must be licensed under Apache 2.0.
- It must handle memory limitations gracefully.
- It must restart if a fatal error occurs.
- It must run on Debian 8.
- It must be accessible over a network.
- It must have an accessible interface.

2.5 Assumptions and dependencies

XZES40-Transformer will be written to interface with an OS agnostic API for any operating-system level operations (e.g., reading and writing from the cache). The application will be portable to new operating systems by writing an OS-specific interface layer and compiling the binary for the given target platform (e.g., Windows or FreeBSD).

XZES40-Transformer will assume that the relevant libraries and languages listed under Software Interfaces are already installed. The installation package we create will resolve these dependencies if they are not already installed with the correct version.

2.6 Apportioning of requirements

Development of the application, user interfaces, and installation packages will be carried out over a 19 weeks, split into three development cycles: Alpha, Beta, and Release.

The Gantt chart can be seen in the Appendix, Figure 4.

2.6.1 Alpha

During the Alpha phase of development we will collect benchmark data, create our basic transformation functionality, and begin work on Cache and Parallel computation optimizations. By the end of week three the application will be able to accept two input documents and output a transformed document. After the initial transformation functionality is complete work on optimizing this process will take place by adding caching and parallel processing to the transformation cycle.

2.6.2 Beta

During the beta phrase of development the XZES40 team will begin work on the HTTP API, the web interface, the Debian package, and further optimizations on the transformation process. Work on the web interface is not possible without the CGI interface first being added, however most other development can take place in parallel.

2.6.3 Release

During the Release phrase we will work exclusively on stretch goals including the CLI interface as well as the RedHat, BSD, and Windows packages. While these goals would be nice to achieve, we understand that there will probably be overflow from the Alpha and Beta phases of development, so we hope to complete all required deliverables well before the release deadline.

3 SPECIFIC REQUIREMENTS

3.1 External interfaces

XSLT40-Transformer will have two user interfaces: a **Web Interface** and a **Commandline Interface (CLI)**.

3.1.1 Web Interface

The website interface for XZES40-Transformer will include of a form with the following fields:

XML File A file upload field for the XML document.

XSLT File A file upload field for the XSLT document.

Output Filename (optional) The filename of the output document. If one is not specified a name will be generated of the following format `document-transform-<date>.xml>`.

The website will make an HTTP `POST` request to the server. This `POST` request will include an XML and XSLT document in it's payload.

The page will redirect the user to a new page where they can download the transformed file.

The Web Interface will require a web-browser (supporting HTTP4.0+).

A prototype of the web interface can be seen in the Appendix, Figure 2.

3.1.2 Commandline Interface (CLI)

The CLI will give users a text-based interface with XZES40-Transformer. The with the following flags:

`-xml-file=` Specifies the input XML document.

`-xslt-file=` Specifies the input XSLT document.

`-server=` Specifies which server to connect to (`e.g., http://servername.ext`)

`-output-file=` (optional) Specifies a file to write out to. Otherwise writes to a file of the following format `document-transform-`

`-port=` (optional) Specifies which port to connect through if non-standard (`e.g., 8001`)

`-help=` (optional) Prints out a help menu (describing these flags)

The CLI will take the following arguments and make a `POST` request to the server. The transformed file will be automatically downloaded to the user's desired location, or to the current working directory with the automatic file name.

The CLI requires a UNIX terminal and UNIX shell.

An example of some CLI interactions can be seen in the Appendix, Figure 3.

Both interfaces XZES40-Transformer will require a method of input (keyboard and mouse or touchscreen), an internet connection, and monitor.

As input both interfaces expect one XML 1.0 formatted document and one XSLT 1.0 formatted document. These files can be UTF-8 or ASCII character encoding.

As output both interfaces will send the user an XML 1.0 formatted document of UTF-8 character encoding.

3.2 Functions

The following functions will be the core functionality of XZES40-Transformer.

- `int transform(string XML_filename, string XSLT_filename, string output_filename)`: This function is called to transform the given XML_FILE with the XSLT_FILE. If output_file is defined the file will be written to that location. If output_file is not defined the new file will be written to STDOUT.

Returns a status macro (SUCCESS or FAILURE).

- `type_cache* get_cache(input_filename)`: Returns a pointer to the cached file.
- `type_cache check_cache(string input_filename)`: The system will check if the given file is in the cache. Returns TRUE if the file is in the cache and FALSE if the file is not in the cache.
- `type_cache set_cache(string input_filename)`: The new XML file will be saved in the cache. Returns SUCCESS or FAILURe macro if the document was or was not successfully cached.
- `int compile(string input_filename)`:
  The system will compile the given XML/XSLT file into machine code to later be transformed.
- `int delete_cache()`: Removes old documents from the cache which are not being used. Triggered when the cache is filled to a certain capacity.

3.3 Performance requirements

XZES40-Transformer will perform better than existing Open Source XML transformation software. The application will have a higher rate transformations per minute, normalizing for input file-size. Given a standard set XML + XSLT document pairs, the application will complete the transformations on average faster than it's leading competitors.

The transformations will also be verified for correctness. It is expected that the application will perform document transformation with high correctness.

3.4 Logical database requirements

XZES40-Transformer application does not require database.

3.5 Design constraints

For the XML compilation and transformation process we are restricted to using the Xerces-C and Xalan-C libraries. For document encoding and decoding we will use the ICU UTF-8 libary.

3.5.1 Standards Compliance

Input documents must be correctly formatted XML and XSLT documents. Malformed documents will be rejected by the application.

Correctly formatted XML documents follow the W3C outlined XML 1.0 and XSLT 1.0 formats. [1] [2]

Our application will also communicate with users over HTTP/HTTPS, however we are not implementing these standards, just using them to communicate over the internet.

3.6 Software System Attributes

3.6.1 Reliability

XZES40-Transformer will be reliable if it's cache is up to date, to avoid transforming documents incorrectly.

3.6.2 Availability

Since XZES40-Transformer will be run as a web-service it should be highly-available during business hours. It will be configured to handle heavy workloads and restart if it crashes. Upon restart it's cache will be cleared.

3.6.3 Security

XZES40-Transformer may be used to handle sensitive data, however it is not this team's job to account for that.

If administrators want the application to be secure they may choose to deploy it behind an organization firewall or configure the web-service to use HTTPS instead of HTTP. The application's HTTP traffic will be handled by Apache via a CGI script, this means that any Apache webserver configuration can be used with XZES40-Transformer's web API.

3.6.4 Maintainability

Our application will be deployed as a daemon which will be configured upon installation. This configuration may be modified after installation, but should work "out of the box".

3.6.5 Portability

As an Open Source project XZES40-Transformer will be designed for portability. It will perform all operating-system specific operations via an OS agnostic API. When the application is compiled on a new platform it will compile against the given OS API (e.g., Windows, Linux, MacOS, etc).

As for installation packages, one must be manually created for each platform, however this is not an urgent requirement and can be carried out platform-by-platform after the initial development is completed.

3.7 Organizing the specific requirements

3.7.1 System Features

3.7.1.1 Transformation

The core feature of XZES40-Transformer will be the XML + XSLT document transformation.

Stimulus:

As input the application will accept two files, one XML 1.0 formatted document and one XSLT 1.0 formatted document.

Response

As output the application will return one XML 1.0 formatted document. This will be the output of the program.

If a mal-formatted document is given as input the output of the program will be an appropriate error.

3.7.1.2 Caching

To increase performance the application will cache certain compiled documents.

Stimulus:

As input the in-memory cache will accept either a compiled file for caching or a hash of a compiled file for retrieval.

Response

If the cache is given a hash and "asked" for a cached document it will either respond with the contents of the compiled file or a FALSE response, signifying the document is not in the cache.

If the cache is given a file for storage it will respond with either a SUCCESS of FAILURE response if the document was successfully cached or not appropriately.

3.7.1.3 Parallel Computation

To further increase performance the application will carry out independent calculations in parallel.

This feature does not have easily defined “stimulus + response” pairs.

The application will need to be designed to avoid race-conditions during parallel computations.

3.7.1.4 Web API

The application will have a web-accessible API.

Stimulus

As input the API will accept either GET or POST requests. POST requests will contain a payload with the XML + XSLT document pair as listed above for the transformation feature.

Response

As response to a GET request the API will respond with a SUCCESS, saying that the application could be reached.

As response to a POST request the application will respond with either a transformed XML document or an appropriate error.

3.7.1.5 User Interface

The application will have two user interfaces. These will allow the user to provide the above stimuli to the web API, download the response file, or display the error message from the API.

3.8 Additional comments

There are no additional comments to be made at this point.

Component	Owner
Document Transformer	Elijah C. Voigt
Website Interface	Shuai Peng
Web API	Elijah C. Voigt
Document Cache	Shuai Peng
Daemon Process	Elijah C. Voigt
Parallel Document Transformation	Zixun Lu
Benchmarking	Zixun Lu
Debian Software Package	Zixun Lu

Figure 1: Required project components ordered by importance

Component	Owner
CLI Interface	Elijah C. Voigt
CentOS Linux software package	Elijah C. Voigt
Windows software package	Shuai Peng
FreeBSD software package	Elijah C. Voigt

Figure 2: Stretch goal project components ordered by importance

4 Supporting Information

The following table outlines **required** project components ordered by importance:

4.1 Appendixes

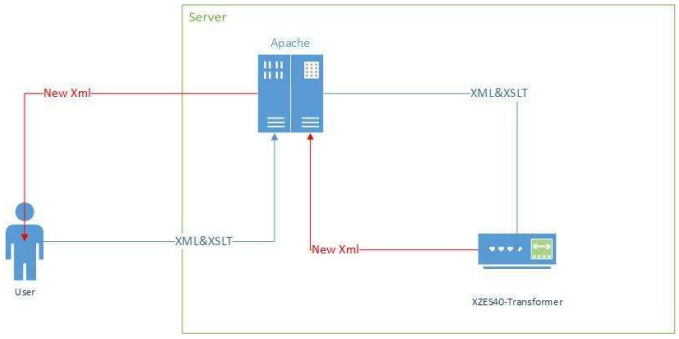

Figure 3: Diagram of dataflow. Documents are sent to the application through the Apache gateway interface. Once processed documents are returned to the user through the gateway interface again.

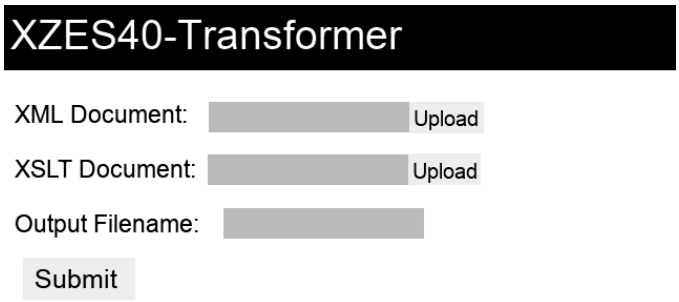

Figure 4: Prototype of the Web Interface. Demonstrates the simplicity of the interface and required form fields.

```
Normal usage:
$ xzes40cli --xml-file='./my-file.xml' --xslt-file='./my-other-file.xslt' --server='http://example.com/
    xzes40-transformer' --output-file='./newfile.xml' --port='8001'
Sending XML and XSLT files to http:/example.com:8001/xzes40-transformer
Transformation complete. Downloading response file to newfile.xml

Sending a bad file:
$ xzes40cli --xml-file='./badfile.jpg' --xslt-file='./badfile.txt' --server='http://example.com/xzes40-
    transformer'
Sending XML and XSLT files to http:/example.com:80/xzes40-transformer
ERROR: Server was unable to transform the requested files.

Using inadequate parameters
$ xzes40cli
Please provide an xml file (--xml-file), xslt file (--xslt-file) and a host (--server).
```

Figure 5: Example use-cases of CLI interface.

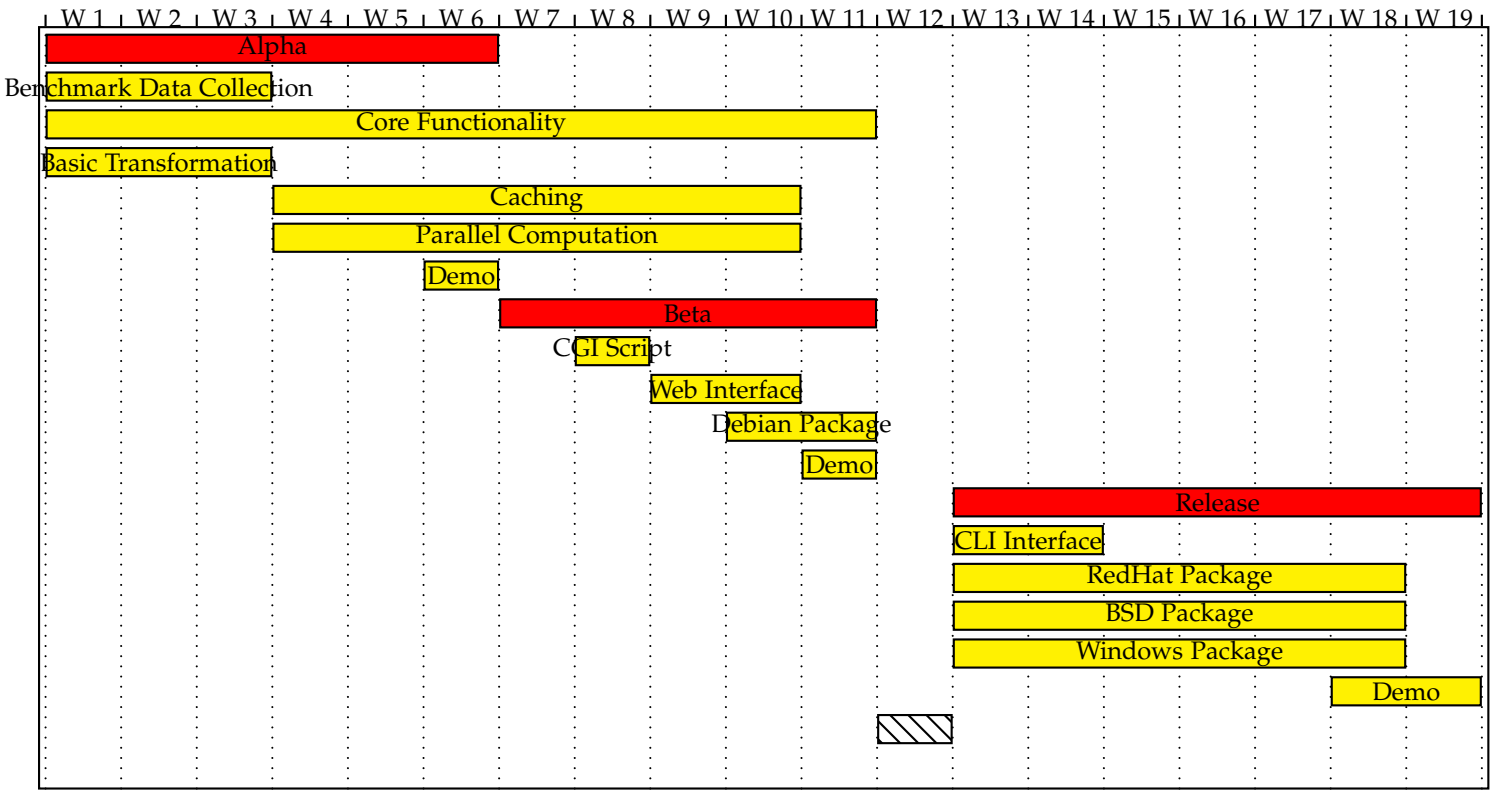

Figure 6: Development Gantt Chart Timeline.

Elijah C. Voigt

Signature

Date

Zixun Lu

Signature

Date

Shuai Peng

Signature

Date

Steven Hathaway

Signature

Date