# Problem Statement

CS 461 - CS Senior Capstone

XZES40 | Group #40

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

November 9, 2016

**Abstract**

## 1 Introduction

## 2 Research and Benchmarking

### 2.1

### 2.2

### 2.3

## 3 XML/XSLT Document Transformation

### 3.1

### 3.2

### 3.3

## 4 XML/XSLT Document Parsing

**Owner: Elijah 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 Xerces-C/C++

The Apache Xerces-C/C++ library is a feature rich XML parsing library. It implements the XML 1.0 standard, it is well documented, and implements DOM as well as SAX methods of document parsing. Most importantly it integrates well with Xalan-C/C++, which is our top choice for performing document parsing.

This library was also explicitly requested by our client, a member of the Apache Software Foundation, so we should have some amount of internal organization support if we need to use it.

### 4.2 DOM Parsing

The first method of XML document parsing is DOM parsing. This method stores the entire parsed XML object in memory to be traversed later on. This operation is memory intensive but is useful for carrying out many operations on a document. [**?**]

### 4.3 SAX parsing

In contrast, SAX document parsing is performed on a data stream. SAX uses callbacks to trigger transformations on an XML document. This method is less memory intensive and can result in a performance boost, but requires a document stream for transformations to happen. [**?**]

## 4.4 Conclusion

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 best. DOM parsing produces an easily cached tree which we can store and retrieve for later operations. It may be worth-while to investigate using SAX parsing early on in development because we find a notable performance boost, so some amount of SAX proof of concept work will be done for the sake of being thorough, but DOM will be the targeted document parsing method.

# 5 XML/XSLT Document Caching

## 5.1

## 5.2

## 5.3

# 6 Parallel Document Transformation

## 6.1

## 6.2

## 6.3

# 7 Web API

#### 7.0.0.1 Owner: Elijah Voigt

## 7.1

## 7.2

## 7.3

## 7.4 Conclusion

# 8 Website

## 8.1

## 8.2

## 8.3

# 9 Debian Package

## 9.1

## 9.2

## 9.3

# 10 Command Line Interface

## 10.1

## 10.2

## 10.3

# 11 Centos Package

## 11.1

## 11.2

## 11.3

# 12 Windows Package

## 12.1

## 12.2

## 12.3

# 13 BSD Package

## 13.1

## 15 References

[?] [?] [?]

## References

[1] *Apache Xalan*. "http://xalan.apache.org.

[2] *Apache Xerces*. http://xerces.apache.org.

[3] *Events vs. Trees*. http://sax.sourceforge.net/event.html.

[4] *International Components of Unicode*. http://icu-project.org.

| | | |
|---|---|---|
| Elijah C. Voigt | Zixun Lu | Shuai Peng |
| *Signature* | *Signature* | *Signature* |
| *Date* | *Date* | *Date* |

Steven Hathaway

*Signature*

*Date*