University of Leicester

SiXML Version 1.2

Succinct indexible XML (SiXML) version 1.2 provides an efficient in-memory representation of large XML documents, with stable and predictable memory usage. It provides a framework around which a variety of XML processing APIs can be implemented. SiXML is based on succinct data structures, whose memory usage is information-theoretically minimum.

http://www.cs.le.ac.uk/SiXML/

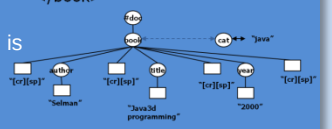
Case Study: Apache Xerces-C++ 2.8

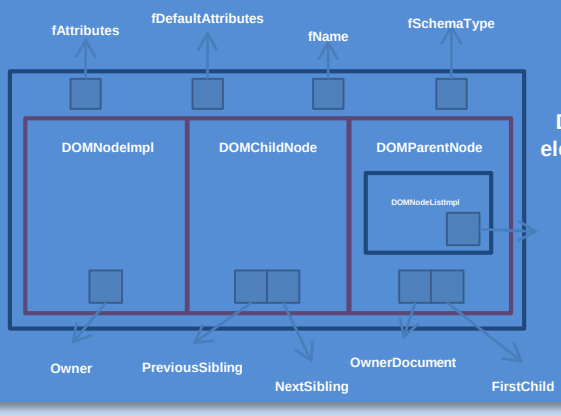
Represents XML documents in main memory as a tree.

- Fast navigation.
- Fast data access/modification.

A Xerces-C++ DOM element node is almost all pointers and requires 416 bits on a 32-bit machine.

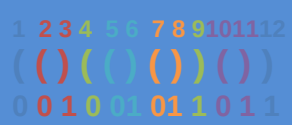
```
<book catalogue="java">
 <author>Selman</author>
 <title>Java3d programming</title>
 <year>2000</year>
</book>
```

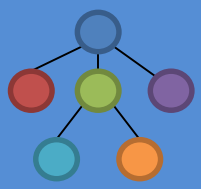




Pointerless Data Structures

- Represent the structure of the tree as a parenthesis string,
- plus efficient index for "matching parens" (total <5 bits/node).
- Also using pointer-less data structures for attributes, text etc.

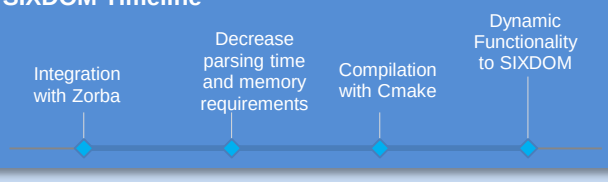




SiXDOM 1.2 Features:

- 64-bit C++ implementation of DOM API (Level 2 and partially 3)
  - Only contains the static DOM methods
- SWIG Bindings for Java and Python

SIXDOM Timeline



SiXDOM and Xerces Performance Comparision

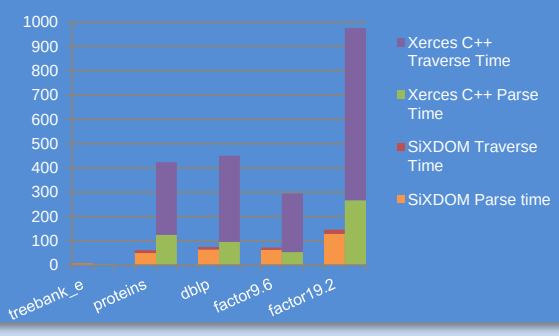
Xerces-C++ vs. SiXDOM (C++) and Xerces2 vs SiXDOM(Java) on benchmark XML documents on Linux PC with 8GB RAM.

- **C++ Virtual Memory Usage**

	Treebank	proteins	dblp	Xmark fact9.6	Xmark fact19.2	Xmark fact38.4
File size (MB)	82	684	840	1073	2150	4300
SiXDOM	127%	95%	110%	100%	95%	95%
Xerces-C++	1350%	1340%	1310%	830%	825%	FAILED

(VM usage given as percentage of file size)

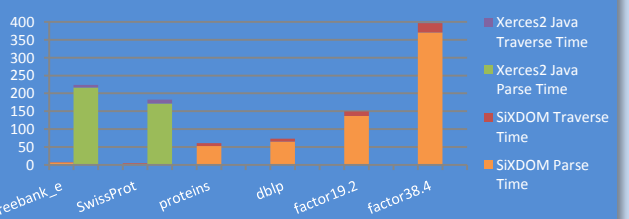
- **C++ Parse and Traverse Speed (Wall Clock Time, secs)**



- **Java Resident Memory Usage**

	Treebank	SwissProt	proteins	dblp	Xmark factor19.2	Xmark factor38.4
File size (MB)	82	109	684	840	2150	4300
SiXDOM	134%	131%	96%	112%	103%	105%
Xerces2 Java	814%	1033%	FAILED	FAILED	FAILED	FAILED

- **Java Parse and Traverse speed (Wall Clock Time, secs)**



Note

- Each SiXDOM document collection has 3-4MB fixed cost.
- Use TreeWalker Interface to avoid memory leaks during traversal (Node objects are created on demand and garbage-collected on deletion of SiXDOM document collection).

We are currently looking for industrial partners.
UK government funding available for joint projects.

Andreas Poyias, Stelios Joannou ({ap468, sj148}@leicester.ac.uk)

SiXDOM developed by O'Neil Delpratt, Stelios Joannou, Naila Rahman and Rajeev Raman