

# XSieve: XSLT+Scheme

## **XSieve**

Extending XSLT with the roots of XSLT

Oleg Parashchenko

Saint-Petersburg State University, Russia

olpa@ <http://uucode.com/blog/>  
<http://xmlhack.ru/>

# Top secret

XSieve is a by-side product.

# Outline

- **XSieve vs alternatives**
- SXML format and tools
- XSieve language
- XSieve in practice
- Technical details
- Conclusion and further work

# Background

My featured skill is:

- XML data transformation
- Word to XML
- Quicksilver, Interleave to XML
- Unstructured FrameMaker to XML
- Etc
  
- XML to HTML, PDF, etc

# XSLT is the best

But XSLT fails.

Data transformation should be supported by programming.

XSLT is not a good programming language.

# Another language?

## "elemname[i]" vs

```
/**
 * Returns child element at given position. Position can be
 * negative. In this case nodes are counting from end.
 * Last node position is '-1'.
 * @param node parent node
 * @param name name of node, null if not important
 * @param pos position of node starting from zero, can be negative
 */
public static Element getChildElement (Element node, String name, int s)
{
    //
    // Update position to be always positive.
    //
    boolean fromEnd = pos < 0;
    if (fromEnd) {
        pos = -(pos + 1);
    }
    //
    // Initialize
    //
    Node cur_node = fromEnd ? node.getLastChild(): node.getFirstChild();
    int cur_pos = -1;
    //
    // Walk on children
    //
```

```
for (; cur_node != null;
    cur_node = (fromEnd
        ? cur_node.getPreviousSibling()
        : cur_node.getNextSibling ()))
{
    //
    // Check that current node is of type 'element'
    //
    if (cur_node.getNodeType() != Node.ELEMENT_NODE) {
        continue;
    }
    //
    // Get element, check its name and position
    //
    Element cur_elem = (Element)cur_node;
    if ((name != null) && (! name.equals (cur_elem.getTagName())) {
        continue;
    }
    cur_pos++;
    if (cur_pos == pos) {
        return cur_elem;
    }
}
//
// If node was found, it was returned from inside loop
//
return null;
}
```

# Good library?

“Isomorphism” to XPath and XSLT

Greenspun's Tenth Rule of Programming:  
“Any sufficiently complicated C or Fortran program contains an ad hoc informally-specified bug-ridden slow implementation of half of Common Lisp”

My addition: the same for data transformation and XSLT.

# The need

XSLT plus a programming language

- XSLT extensions?
- New XML languages?
- Old languages?

# DSSSL

Scheme dialect, but not Scheme.

Genealogy:



# SXML library

- Scheme
- XPath, XSLT and more
- Effective design
- A separate sandbox

# What XSieve fixes

XSLT      A general-purpose  
programming language

SXML      Integration with XSLT

# Gestalt entity

“A gestalt entity is a physical, biological, psychological, or symbolic configuration or pattern of elements, so unified as a whole that its properties cannot be derived from a simple summation of its parts” — Wikipedia

# Outline

- XSieve vs alternatives
- **SXML format and tools**
- XSieve language
- XSieve in practice
- Technical details
- Conclusion and further work

# SXML format

“Show me your code and conceal your data structures, and I shall continue to be mystified. Show me your data structures, and I won't usually need your code; it'll be obvious”

Eric Raymond (1997), paraphrase of Frederick Brooks (1975)

# SXML format

`<elem>text</elem>`

`(elem "text")`

`<elem>text<sube>subtext</sube></elem>`

`(elem "text" (sube "subtext"))`

# SXML format

```
<?pi db as-table?>  
(*PI* db "as-table")
```

```
<!--TODO-->  
(*COMMENT* "TODO")
```

```
(*TOP* (... ))
```

# SXML format

```
<elem a1="val1" a2="val2" />
```

```
(elem (@  
  (a1 "val1")  
  (a2 "val2")))
```

# Navigation by hand

```
x = (elem "a" (b) "c")  
      <elem>a<b/>c</elem>
```

```
(car x) == name()  
(cdr x) == @* | node()
```

```
(map func (query x)) ~ xsl:apply-templates
```

# SXPath

(elem1 elem2 @ attr)  
elem1/elem2/@attr

(elem1 (elem2 condition) @ attr)  
elem1/elem2[condition]/@attr

Custom Scheme steps, axes, predicates

# SXSLT

- Local scoping of re-writing “templates”
- First-class stylesheets
- Traversal strategies
- Ability to re-traverse the original or transformed trees

# Outline

- XSieve vs alternatives
- SXML format and tools
- **XSieve language**
- XSieve in practice
- Technical details
- Conclusion and further work

# XSieve example

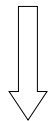
```
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:s   = "http://xsieve.sourceforge.net"
  extension-element-prefixes="s"
  version   = "1.0">

  <xsl:template match="/">
    <s:scheme>
      (display (x:current))(newline)
    </s:scheme>
  </xsl:template>

</xsl:stylesheet>
```

# XSieve example

```
<xsl:template match="/">
  <s:scheme>
    '(article (@ (id "hw"))
      (title "Hello")
      (para "Hello, " (object "World") "!"))
  </s:scheme>
</xsl:template>
```



```
<article id="hw">
  <title>Hello</title>
  <para>Hello, <object>World</object>!</para>
</article>
```

# More XSieve

```
(x:eval xpath [basenode])
```

```
(x:apply-templates  
  ['mode 'mode-name]  
  ['with-param 'param1 value1 ...]  
  ['with-param 'paramN valueN]  
  nodeset)
```

# Multiple sum

```
<items>
  <item price="20" qty="2"/>
  ...
  <item price="50" qty="0"/>
</items>
```

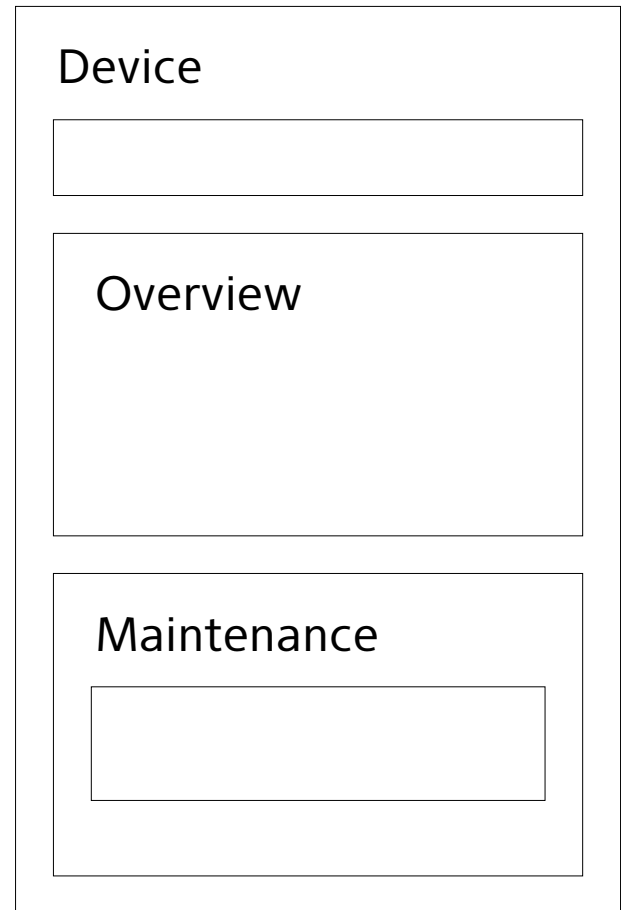
```
(apply +
 (map (lambda (node)
  (* (x:eval "number(@qty)" node)
    (x:eval "number(@price)" node)))
 (x:eval "//item")))
```

# Outline

- XSieve vs alternatives
- SXML format and tools
- XSieve language
- **XSieve in practice**
- Technical details
- Conclusion and further work

# (Re)grouping

```
<h2>Device</h2>  
<p>In this section...</p>  
<h3>Overview</h3>  
...  
<h3>Maintenance</h3>  
<p>1. ...</p>  
<p>2. ...</p>  
...
```



# OpenOffice hinting

```
<text:p text:style-name="P9">Some text</text:p>
```



```
<text:p text:style-name="P9" fo:font-weight="bold"  
><style-hint:name>P9</style-hint:name  
><style-hint:name>Standard</style-hint:name  
>Some text</text:p>
```

# Syntax highlighting

```
<programlisting role="xml">
&lt;para>Hello, <emphasis>&amp;who;</emphasis
>!&lt;/para> <co id="who-entity"/>
</programlisting>
```

Default:        <para>Hello, &who;!  
</para> (1)

Highlighted: <para>Hello, &who;!  
</para>

XSieve:        <para>Hello, &who;!  
</para> (1)

<http://tohtml.com/dbsy/>

# Outline

- XSieve vs alternatives
- SXML format and tools
- XSieve language
- XSieve in practice
- **Technical details**
- Conclusion and further work

# The executable

- Standard XSLT extension
- Scheme as the language
- xsltproc plus Guile
- xsltproc plugin

# The main problems were

- Uncertainty with results
- S-expressions and XML are different, incompatible creatures

# Other problems were

- XML  $\Leftrightarrow$  SXML: namespaces
- Garbage collection and memory management
- apply-templates from "s:scheme"

# Correctness

DocBook stylesheets

```
<xsl:apply-templates select="xpath"/>
```

```
<s:scheme>
```

```
  (x:apply-templates (x:eval "xpath"))
```

```
</s:scheme>
```

# Performance

It depends.

- Auto-generated: slow
- OpenOffice hinting: very fast
- Lazy instantiation: very slow

# More implementations?

Java, .NET: problems are easier.

XSieve: alternative or partner for EXSLT.

# Outline

- XSieve vs alternatives
- SXML format and tools
- XSieve language
- XSieve in practice
- Technical details
- **Development behind XSieve**
- Conclusion and further work

# Generative XPath

- Compilers
- Text processors
- Etc
  
- Hard for C
- XML virtual machine as Scheme plus SXML

# Testing XPath

Test suite vs test-by-practice

`(x:apply-templates (x:eval "xpath"))`

# Main development

- <http://xmlhack.ru/protva/xquery/>
- Testing XML VM in C:
  - GNU find with XPath
  - libxml/libxslt
- Testing generative XPath

by-side: XSieve

# Outline

- XSieve vs alternatives
- SXML format and tools
- XSieve language
- XSieve in practice
- Technical details
- **Conclusion and further work**

# Future work

- XSieve is frozen
- DocBook XSieve/TeXML stylesheets

# New language

- What does it look like?
  - Scheme and SXML in XSLT
- What does it fix?
  - XML to programming language
  - SXML integration
- Will I use it?

# Thank you!

## **XSieve**

Extending XSLT with the roots of XSLT

Oleg Parashchenko

Saint-Petersburg State University, Russia

olpa@ <http://uucode.com/blog/>  
<http://xmlhack.ru/>