xproc.xq

Jim Fuller, MarkLogic 2013

Architecture of an xproc processor

James Fuller

Senior engineer

http://jim.fuller.name

http://exslt.org

http://www.xmlprague.cz

@xquery

@perl6

Perlmonks

Pilgrim

SLT UK 2001

Overview

- Why XProc?
- xproc.xq project
- xproc.xq Architecture
- Points of Interests
- Summary

XProc Overview

Xproc Goals

- The language must be expressed as declarative XML and be rich enough to address practical interoperability concerns but also beconcise
- The language must allow the inputs, outputs, and other parameters of a components to be specified with information passed between steps using XML
- The language must define the basic minimal set of processing options and associated error reporting options required to achieve interoperability.
- Given a set of components and a set of documents, the language must allow the order of processing to be specified.
- Agnostic in terms of parallel, serial or streaming processing
- The model should be extensible enough so that applications can define new processes and make them a component in a pipeline.
- The model could allow iteration and conditional processing which also allow selection of different components as a function of run-time evaluation.

Make xml pipelines

Xproc Refresher

```
(<root/>,<root/>,<test/>)
```

```
<p:pipeline version=“1.0”  name="main">
    <p:count/>
</p:pipeline>
```

```
<c:result>3</c:result>
```

Show simple xproc

That’s not quite the whole story …

```
<p:declare-step version='1.0’  name="main">
    <p:input port="source"/>
    <p:output port="result”/>
    <p:count name=”step1”/>
</p:declare-step>
```

Really, not at all…

```
<p:declare-step name="main" xmlns:p="http://www.w3.org/
ns/xproc" version="1.0">
<p:input port="source"/>
<p:output port="result">
    <p:pipe step="step1" port="result"/>
</p:output>
<p:count name="step1">
    <p:input port="source">
        <p:pipe step="main" port="source"/>
    </p:input>
</p:count>
</p:declare-step>
```

Why Xproc ?

- ‘pipes’ are a natural idiom for xml processing
- Flow based versus FSM (draw diagram)
- State is the enemy of dynamic computation
- Complex workflow still possible but YAGNI
- Main control loop

Current news

- http://mesonet.info/
- http://code.google.com/p/daisy-pipeline/wiki/XProcOverview-
- http://balisage.net/Proceedings/vol8/html/Williams01/BalisageVol8-Williams01.html
- https://github.com/gimsieke/epubcheck-xproc
- https://github.com/josteinaj/xprocspec

Current news

W3C XML Processing WG working on Xproc vnext

1. Improve ease of use (syntactic improvements)
2. Increase the scope for working with non XML content
3. Address known shortcomings in the language
4. Improve relationship with streaming and parallel processing

Fix params, non xml doc processing, drop Xpath 1.0, let options+variables contain fragments, allow AVT

xproc.xq

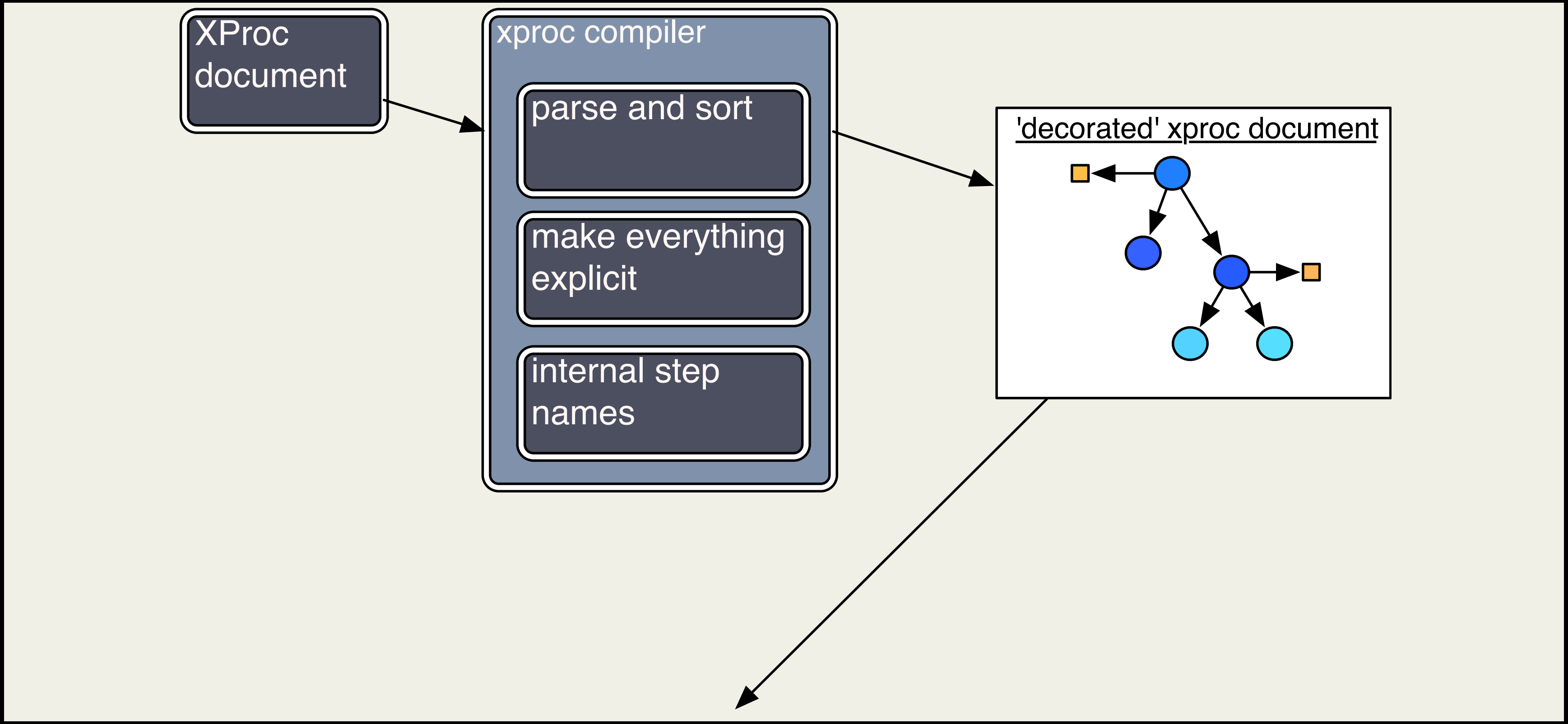
xproc.xq Project

- Xproc processor built with Xquery 3.0 on MarkLogic
- Github Project - https://github.com/xquery/xproc.xq
- Build/test system
  - xray
  - run w3c unit test suite
- dist layout
  - compact
  - extensible
- Xquery entry points
  - flags

Architecture

- Parse: consume and parse Xproc pipeline
- Dynamic evaluation: runtime engine
- Serializer: output results

Parsing


XProc document
xproc compiler
parse and sort
make everything explicit
internal step names
'decorated' xproc document

Decorated pipeline

demo

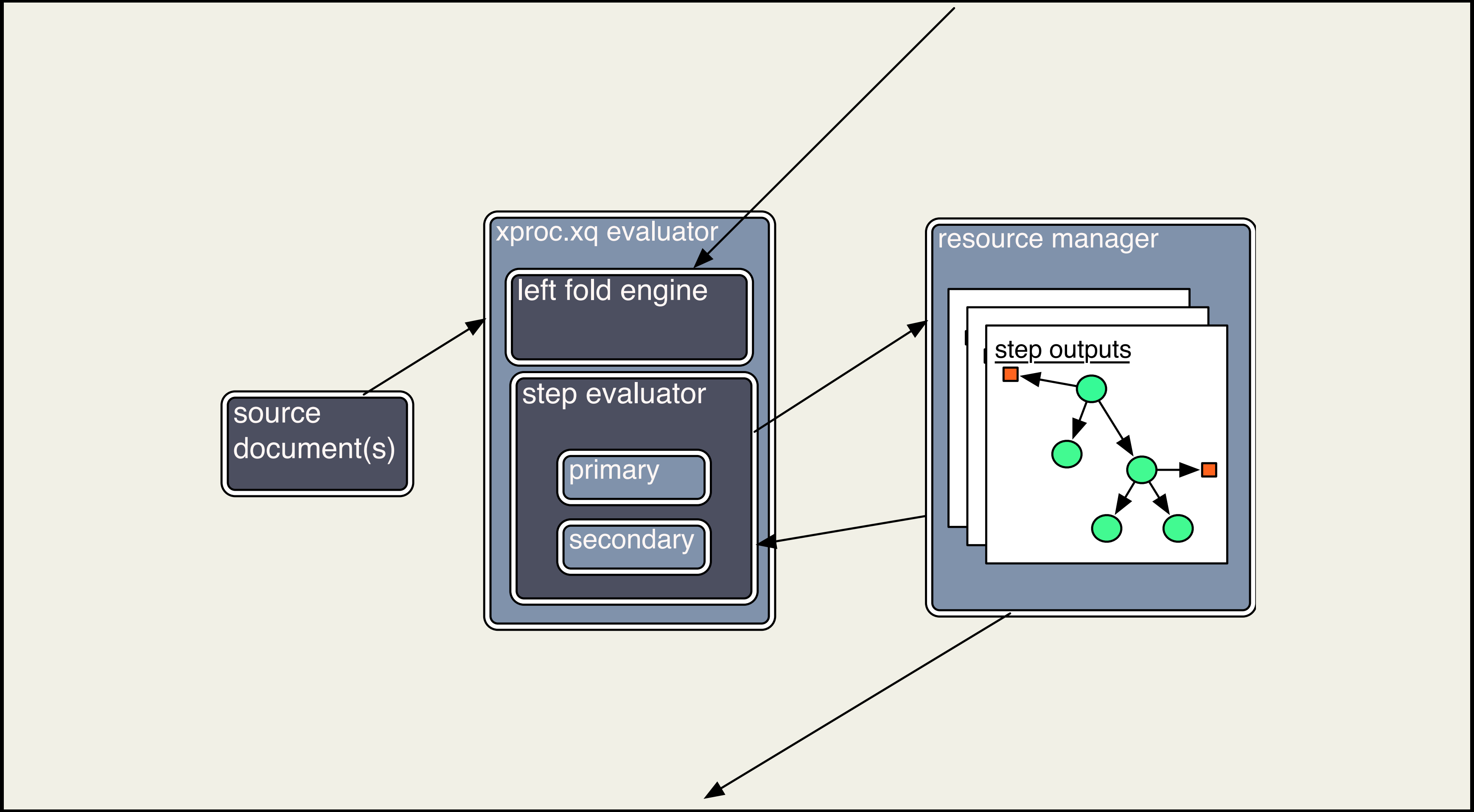
Static analysis - unordered

```
<p:declare-step version='1.0' name="main">
<p:input port="source"/>
<p:output port="result">
    <p:pipe step="i1" port="result"/>
</p:output>

<p:identity>
    <p:input port="source">
        <p:pipe step="main" port="source"/>
    </p:input>
</p:identity>
<p:identity name="i3"/>
<p:identity name="i1">
    <p:input port="source">
        <p:pipe step="i3" port="result"/>
    </p:input>
</p:identity>
</p:declare-step>
```

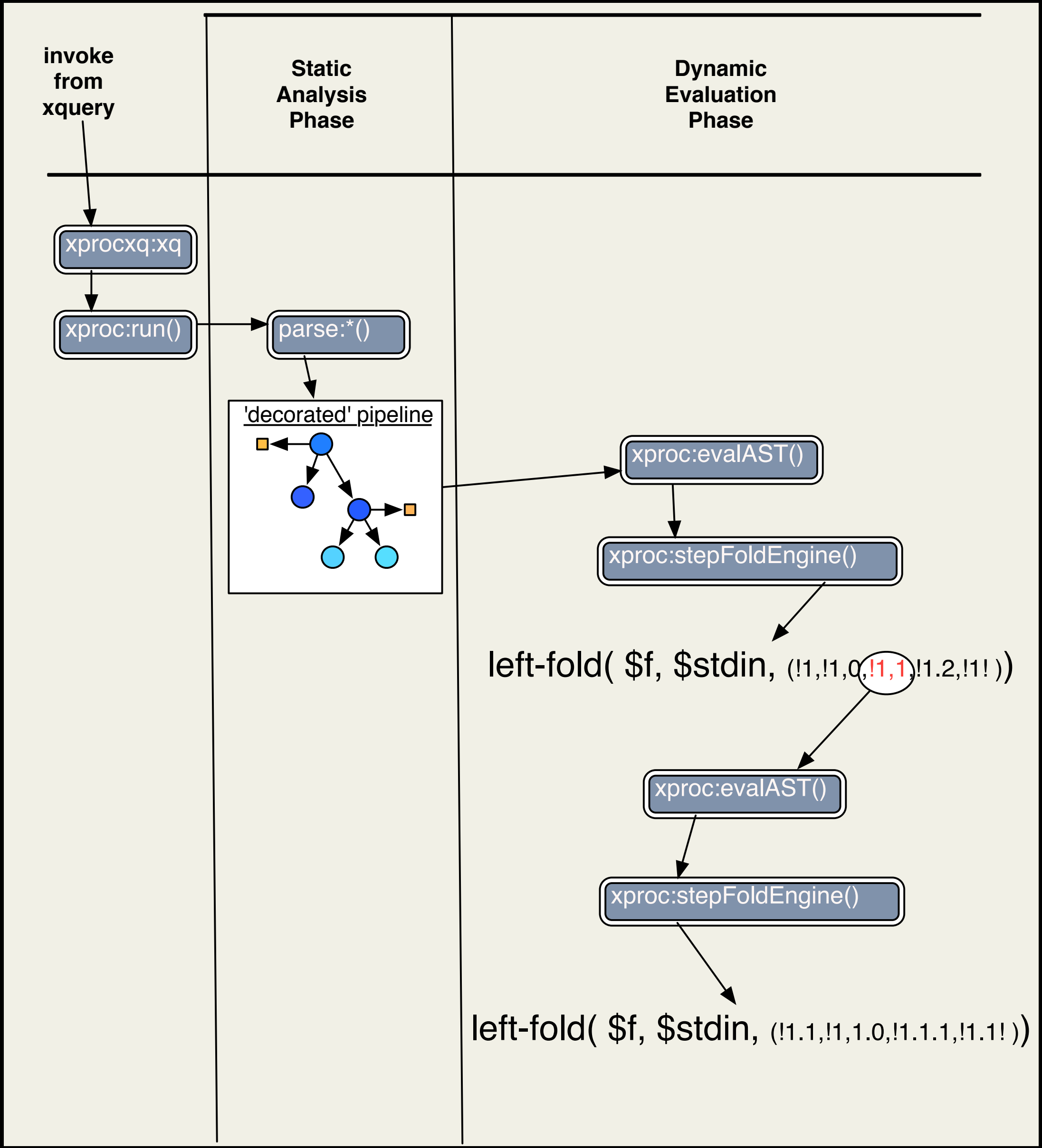
Static analysis - ordered

```
<p:declare-step version='1.0' name="main">
<p:input port="source"/>
<p:output port="result">
    <p:pipe step="i1" port="result"/>
</p:output>

<p:identity>
    <p:input port="source">
        <p:pipe step="main" port="source"/>
    </p:input>
</p:identity>
<p:identity name="i3"/>
<p:identity name="i1">
    <p:input port="source">
        <p:pipe step="i3" port="result"/>
    </p:input>
</p:identity>
</p:declare-step>
```

Runtime

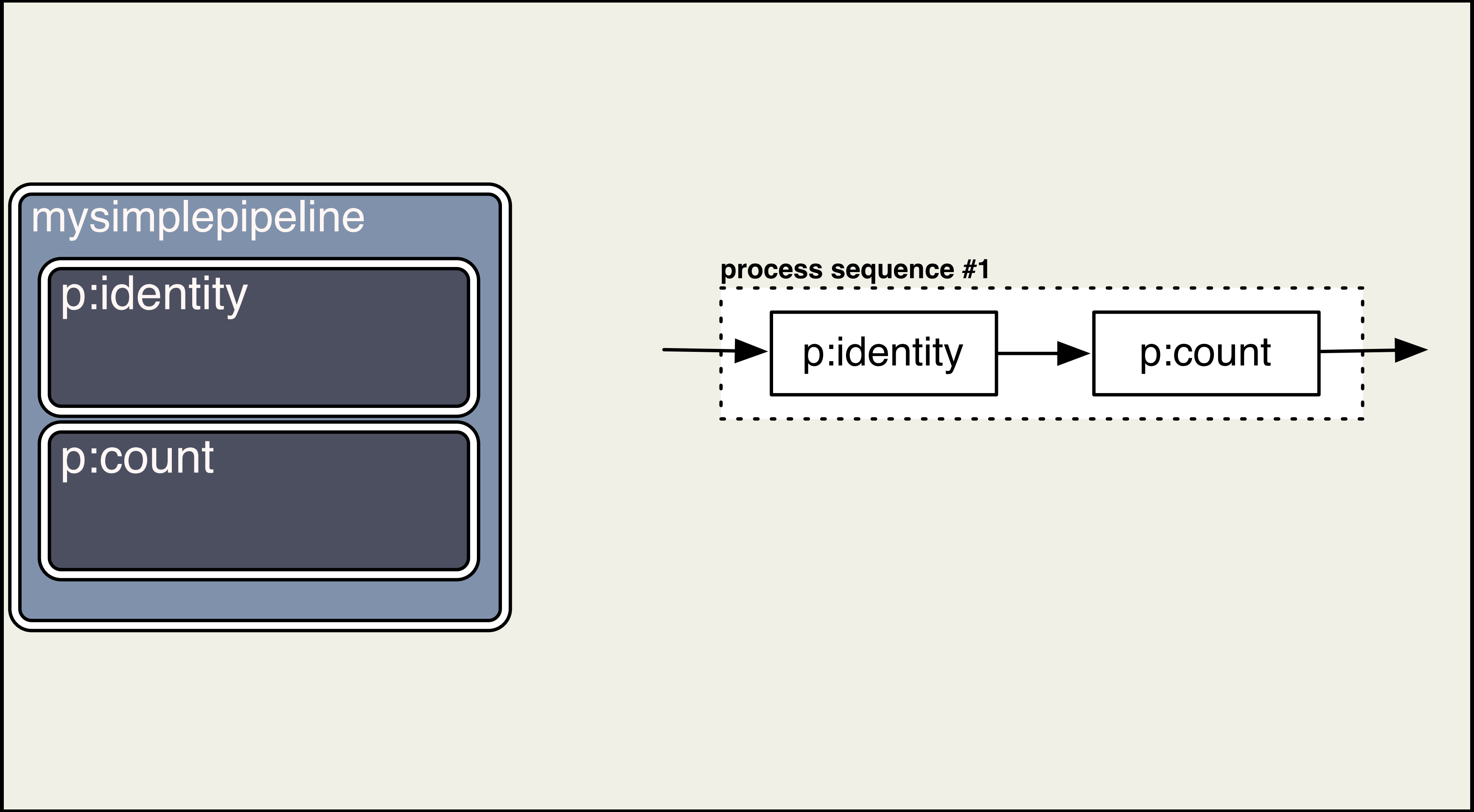

xproc.xq evaluator
left fold engine
step evaluator
primary
secondary
source document(s)
resource manager
step outputs

Runtime

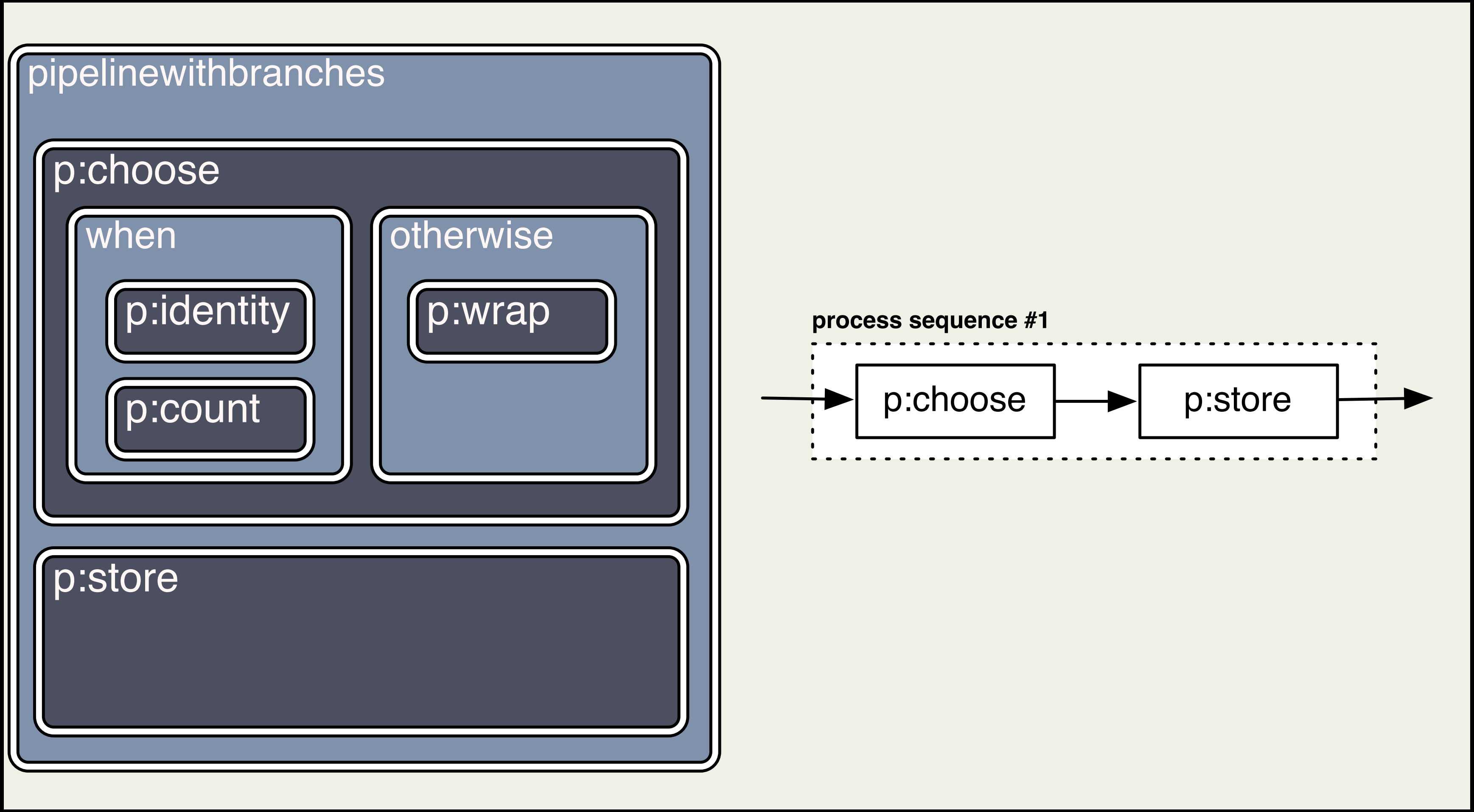

invoke from xquery
Static Analysis Phase
Dynamic Evaluation Phase
xprocxq:xq
xproc:run()
parse:*()
'decorated' pipeline
xproc:evalAST()
xproc:stepFoldEngine()
left-fold($f, $stdin, (!1,!1,0,!1,1,!1.2,!1!))
xproc:evalAST()
xproc:stepFoldEngine()
left-fold($f, $stdin, (!1.1,!1,1.0,!1.1.1,!1.1!))

TIMECHECK

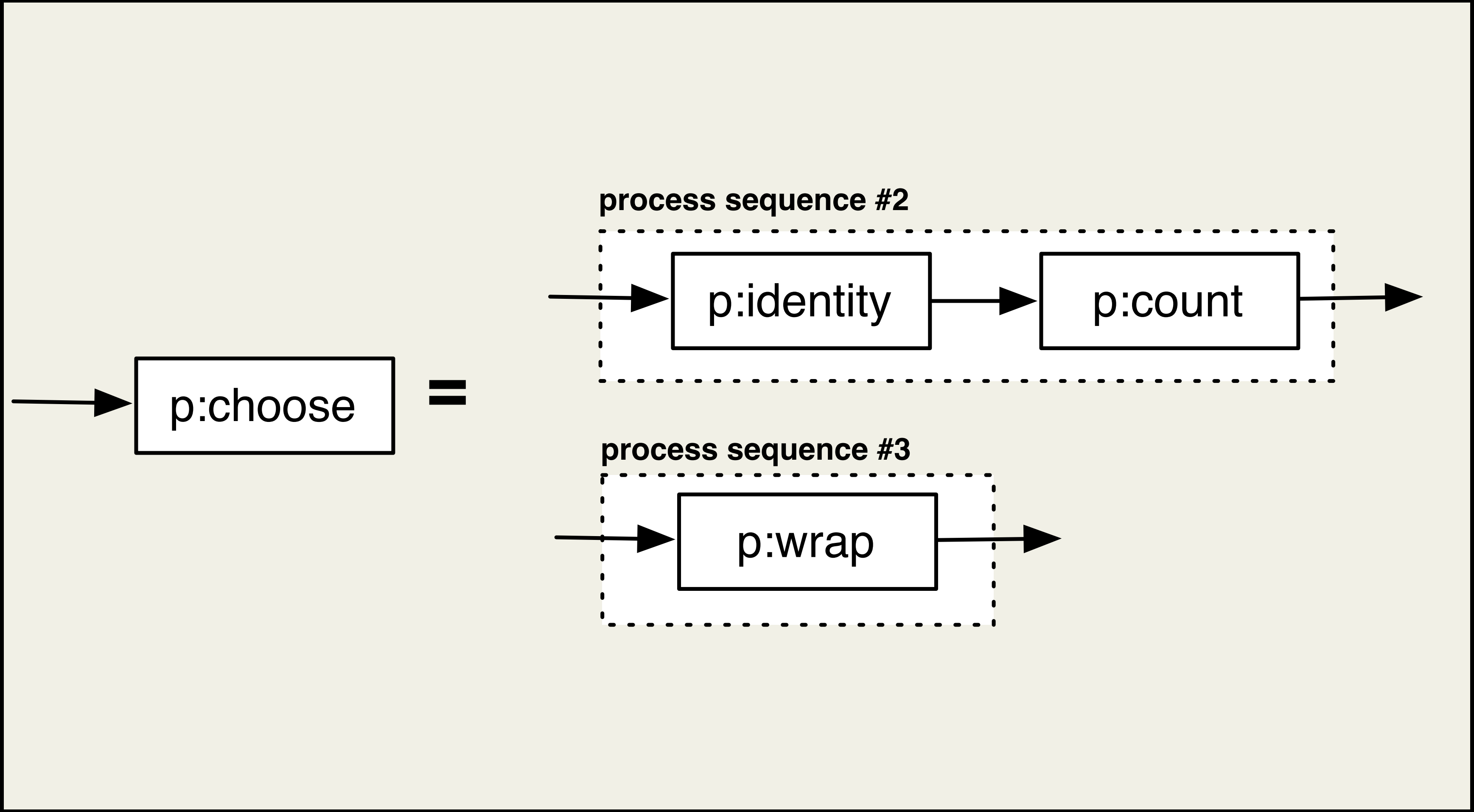
Pipeline decomposition



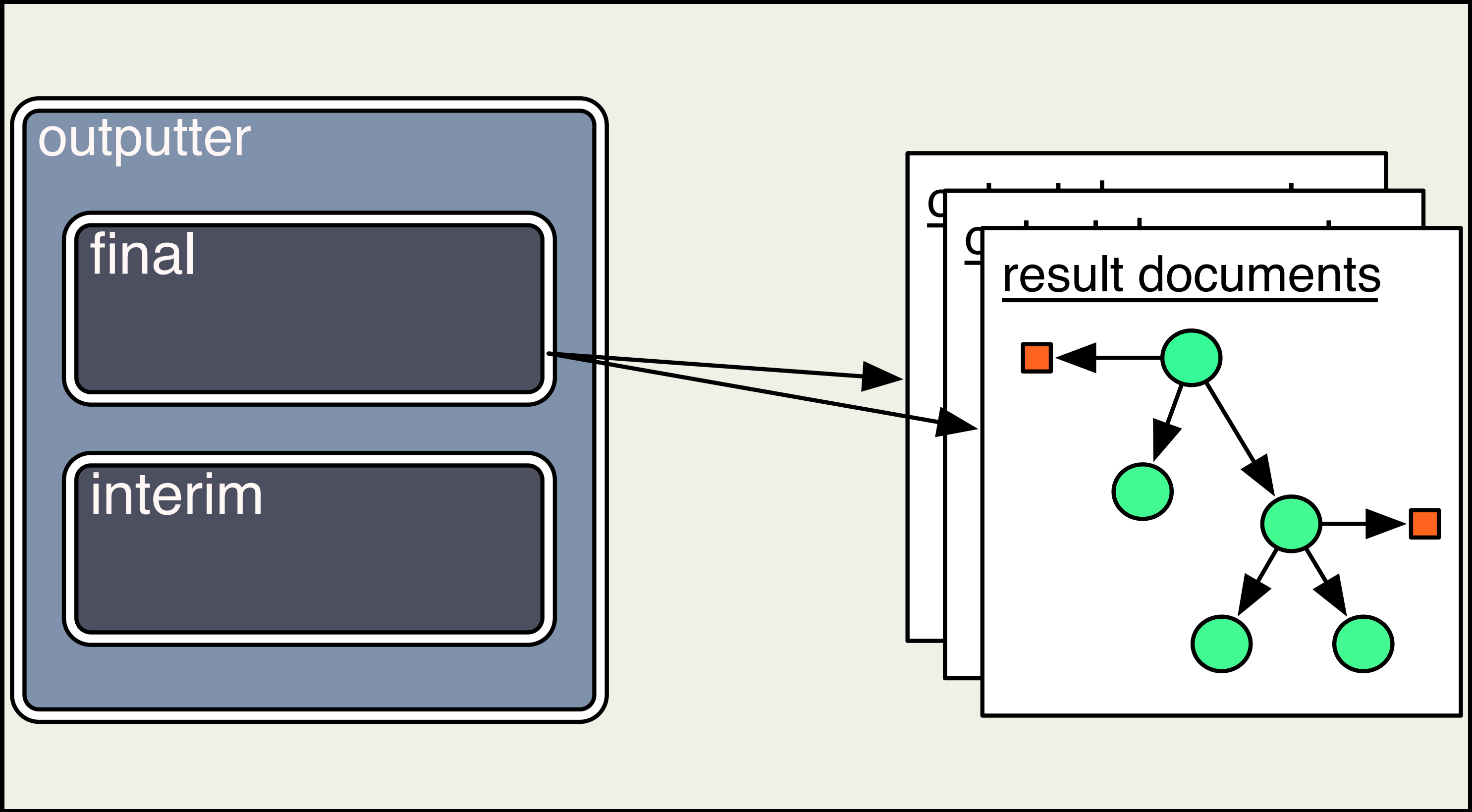
Pipeline decomposition



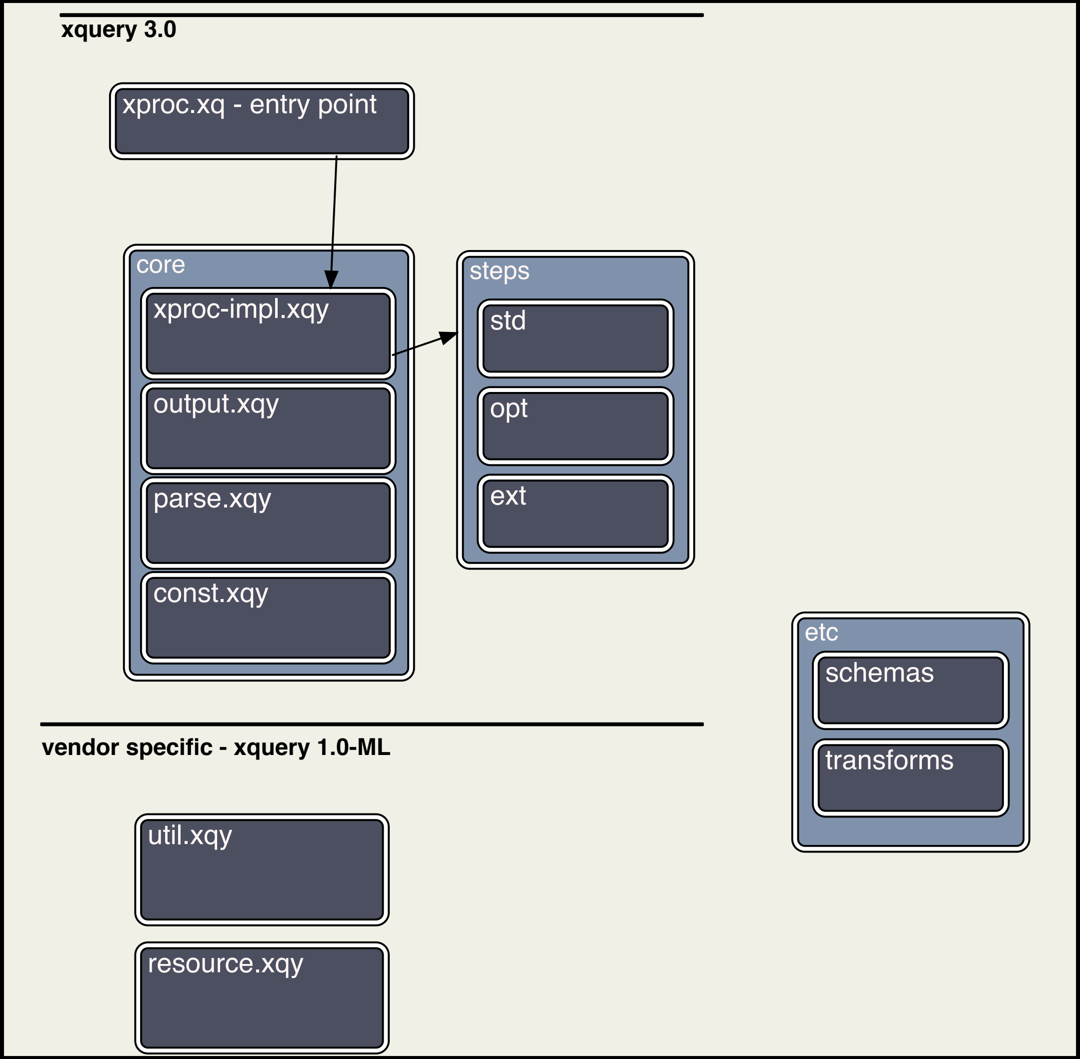
Pipeline decomposition



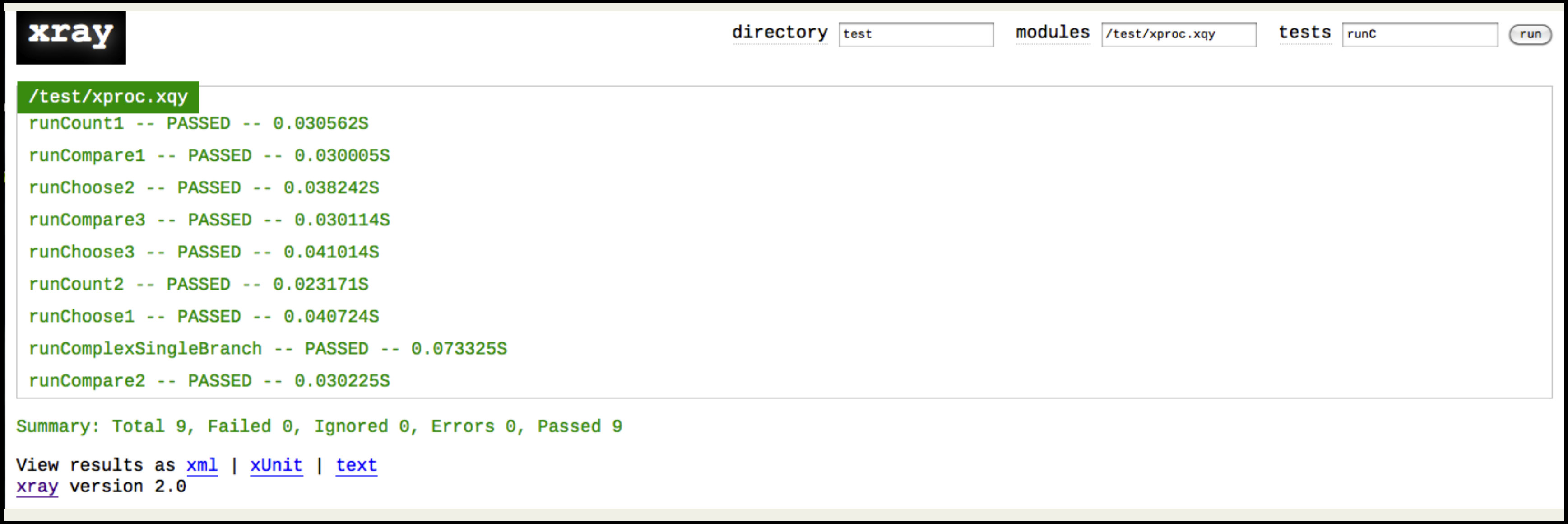
Serializer



File Layout


xquery 3.0
xproc.xq - entry point
core
xproc-impl.xqy
output.xqy
parse.xqy
const.xqy
steps
std
opt
ext
etc
schemas
transforms
vendor specific - xquery 1.0-ML
util.xqy
resource.xqy

XRAY testing



```
/test/xproc.xqy
runCount1 -- PASSED -- 0.030562S
runCompare1 -- PASSED -- 0.030005S
runChoose2 -- PASSED -- 0.038242S
runCompare3 -- PASSED -- 0.030114S
runChoose3 -- PASSED -- 0.041014S
runCount2 -- PASSED -- 0.023171S
runChoose1 -- PASSED -- 0.040724S
runComplexSingleBranch -- PASSED -- 0.073325S
runCompare2 -- PASSED -- 0.030225S

Summary: Total 9, Failed 0, Ignored 0, Errors 0, Passed 9

View results as xml | xUnit | text
xray version 2.0
```

points of interest

Xquery 3.0 to the rescue

- Using a Reducer, such as left-fold(), in combination with dynamic function calls underpin the heart of xproc.xq dynamic evaluation engine.

- XQuery 3.0 annotations feature is employed to identify in the codebase step functions, making it straightforward to author new steps in pure XQuery.

- The choice of the 'flow' work flow model is a perfect match for a functional programming language which has functions as first class citizens. All step inputs and outputs are written once and never mutated thereafter. Changing state 'in-place' is destructive and can represent a loss of fidelity.

Steps with XSLT

- ‘XSLT's polymorphism and dynamic dispatch makes static analysis difficult.’ – Mkay 2009
- Spent many years pipelining XSLT
- XProc dependency on XSLT match patterns combined with the fact that many of the steps lent themselves to implementation using XSLT v2.0,

BYOSR

demo

morefun with a fold engine

- Graph out steps
- Journaling/Logging … etc (inject in ML properties)
- Architectural side effects = powerful runtime idiom

Extensibility and reuse

- Create new step libs at xproc level
- Easily create custom xproc libs from xquery
- Use steps in your own xquery programs as functions

Create new extension step

1. Add src/steps/ext.xqy

2. Add src/extensions/pipeline-extensions.xml

Invoke xproc step in XQuery

```
xquery version "3.0";
import module namespace std = "http://xproc.net/xproc/std" at "/
xquery/steps/std.xqy";

declare namespace p="http://www.w3.org/ns/xproc";
declare namespace xproc = "http://xproc.net/xproc";

std:count( (<test><a>test</a></test>, <test/>),
     (),
     <xproc:options>
       <p:with-option name="limit" select="1000"/>
     </xproc:options>,
     ())
```

summary

Review

- XProc has many favorable characteristics
- XProc getting better
- xproc.xq will get better

The Future

- Support other xquery engines (Saxon …)
- Deeper integration, better compliance
- Analyze performance in database
- CXAN integration

References

[apache-lic-v2] [Apache v2 License] Apache v2.0 License - http://www.apache.org/licenses/LICENSE-2.0.html

[xml-calabash] [XML Calabash] Norm Walsh's XProc processor XML Calabash - http://xmlcalabash.com/

[EXIST] eXist XML Database - http://exist.sourceforge.net

[fsm] [Finite State Machine] Finite State Machine (FSM) entry at wikipedia - https://en.wikipedia.org/wiki/Finite- state_machine

[saxon] [SAXON] Michael Kay's XSLT & XQUERY Processor - http://www.saxonica.com [transform-xq] [transform.xq] John Snelson's transform.xq - https://github.com/jpcs/transform.xq [MarkLogic] MarkLogic - http://www.marklogic.com

[kay2010] [Kay2010] Kay, Michael. "A Streaming XSLT Processor." Presented at Balisage: The Markup Conference 2010, Montréal, Canada, August 3 - 6, 2010. In Proceedings of Balisage: The Markup Conference 2010. Balisage Series on Markup Technologies, vol. 5 (2010). doi:10.4242/BalisageVol5.Kay01. - http://www.balisage.net/Proceedings/vol5/html/Kay01/ BalisageVol5-Kay01.html

[xmlprocwg] [XMLPROCWG] XML Processing Working Group - http://www.w3.org/XML/Processing/

[xml-catalog] [XML-CATALOG] XML catalog - http://en.wikipedia.org/wiki/XML_Catalog

[xproc-spec] [XProc] XProc: An XML Pipeline LanguageW3C Recommendation 11 May 2010 - http:// www.w3.org/TR/xproc

[xproc-use-case-note] [XProc Requirements] XProc XML Processing Model Requirements W3C Working Group Note 05 April 2004 - http://www.w3.org/TR/2004/NOTE-proc-model-req-20040405/

29xproc.xq - Architecture of an XProc processor

[xproc-member-submission-xpl] [XPL] XML Pipeline Language (XPL) Version 1.0 (Draft) W3C Member Submission 11 April 2005 - http://www.w3.org/Submission/xpl/

[xproc-use-cases] [XProc Use Cases] XProc Use Cases - http://www.w3.org/TR/xproc-requirements/

[xprov-vnext-requirements] [XProc vnext] XProc vnext language requirements - http://www.w3.org/XML/XProc/ docs/langreq-v2.html

[known-impl] [XProc Implementations] Known Implementations - http://xproc.org/implementations/

[XPROC-PARALLELISM] [XPROC- PARALLELISM] XProc specification H. Sequential Steps, parallelism, and side-effects - http://www.w3.org/TR/xproc/#parallelism

[XPROC-TEST-SUITE] [XPROC-TEST-SUITE] XProc Test Suite - http://tests.xproc.org/

[xray] [XRAY] - Rob Whitby's XRay - https://github.com/robwhitby/xray

[xproc-system-properties] [XPROC-SYSTEM-PROPERTIES] XProc system properties - http://www.w3.org/TR/ xproc/#f.system-property

[xslt-2] [XSLT 2.0] XSL Transformations (XSLT) Version 2.0. Michael Kay, editor. W3C Recommendation. 23 January 2007. - http://www.w3.org/TR/xslt20/

[xproc.xq] [xproc.xq project] xproc.xq github - https://github.com/xquery/xproc.xq

[zergaoui2009] [Zergaoui2009] Mohamed Zergaoui. Memory management in streaming: Buffering, lookahead, or none. Which to choose? Int Symp on Processing XML Efficiently. 10 Aug 2009, Montreal, Canada. Balisage Series on Markup Technologies, vol. 4 (2009). doi: 10.4242/BalisageVol4.Zergaoui02. - http:// www.balisage.net/Proceedings/vol4/html/Zergaoui02/BalisageVol4-Zergaoui02.html