A Tutorial on XProc

An XML Pipeline Language

Rui Lopes
LaSIGE/University of Lisbon
rlopes@di.fc.ul.pt

Outline

- Pipeline Concepts
- Syntax Overview
- Steps
- Other Pipeline Elements
- Standard Step Library
- Recipes

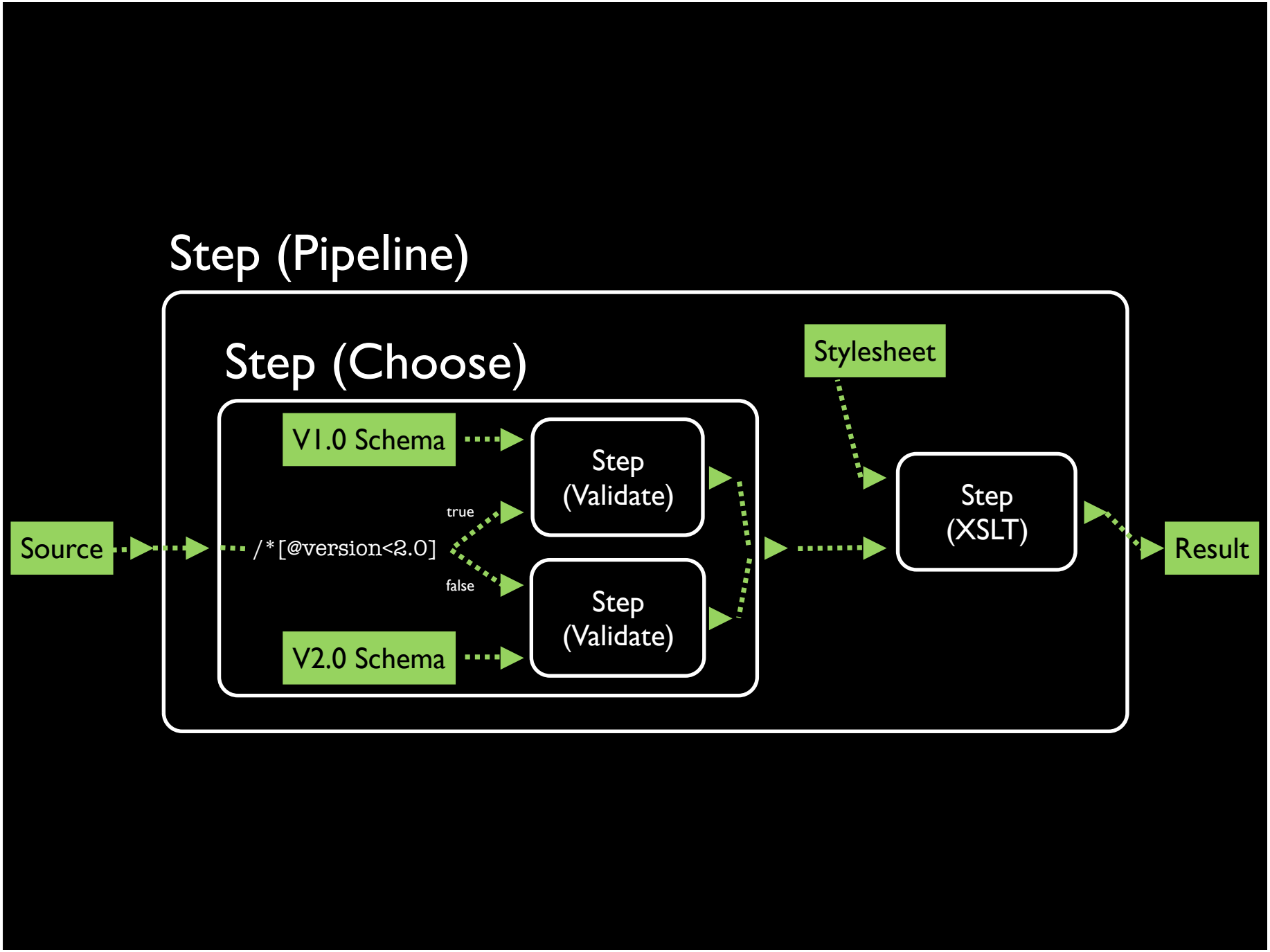
XProc: Background

- XML processing is becoming mainstream
  - XSLT, XQuery, Schema validation, etc.
- How to glue them all coherently, in order to create (complex) XML applications?

XML Pipelines.

http://www.w3.org/XML/XProc/docs/langspec.html

XProc: Concepts


Step (Pipeline)
Step (Choose)
V1.0 Schema
Step
(Validate)
true
Source
/*[@version<2.0]
false
Step
(Validate)
V2.0 Schema
Stylesheet
Step
(XSLT)
Result

Concepts: Steps

- Compound (e.g., Pipeline, Choose)
  - Containers for other steps (defined through subpipelines)
- Atomic (e.g., Validate, XSLT)
- Referenced by name

Concepts: I/O

- XML documents (i.e., infosets) flow inside a pipeline
- A step definition states which inputs and outputs it requires (single or sequence)
- Steps are connected together through pipes between inputs and outputs, defining a pipeline's flow
  - No loops (directed acyclic graph)
- As streamable as possible (huge datasets)

Concepts: I/O

- Steps have primary inputs and outputs (and may have secondary)
- Steps may accept options (name/value pairs)
- Steps may accept parameters (name/value pairs)

Concepts: XPath context

- May occur in several places:
  - Compound steps, compute option and parameter values, values passed to steps
- XPath 1.0 or 2.0 (depending on implementations)

Concepts: XPath context

- Processor vs. step context levels
- XPath extension functions
  - System properties
  - Step available
  - Iteration position
  - Iteration size

XProc: Syntax

```
<p:pipeline xmlns:p="http://www.w3.org/ns/xproc">
    <p:choose>
        <p:when test="/*[@version < 2.0]">
            <p:validate-with-xml-schema>
                <p:input port="schema">
                    <p:document href="v1schema.xsd"/>
                </p:input>
            </p:validate-with-xml-schema>
        </p:when>

        <p:otherwise>
            <p:validate-with-xml-schema>
                <p:input port="schema">
                    <p:document href="v2schema.xsd"/>
                </p:input>
            </p:validate-with-xml-schema>
        </p:otherwise>
    </p:choose>

    <p:xslt>
        <p:input port="stylesheet">
            <p:document href="stylesheet.xsl"/>
        </p:input>
    </p:xslt>
</p:pipeline>
```

```
<p:pipeline xmlns:p="http://www.w3.org/ns/xproc">
    <p:choose>
        <p:when test="/*[@version < 2.0]">
            <p:validate-with-xml-schema>
                <p:input port="schema">
                    <p:document href="v1schema.xsd"/>
                </p:input>
            </p:validate-with-xml-schema>
        </p:when>

        <p:otherwise>
            <p:validate-with-xml-schema>
                <p:input port="schema">
                    <p:document href="v2schema.xsd"/>
                </p:input>
            </p:validate-with-xml-schema>
        </p:otherwise>
    </p:choose>

    <p:xslt>
        <p:input port="stylesheet">
            <p:document href="stylesheet.xsl"/>
        </p:input>
    </p:xslt>
</p:pipeline>
```

```
<p:pipeline xmlns:p="http://www.w3.org/ns/xproc">
    <p:choose>
        <p:when test="/*[@version < 2.0]">
            <p:validate-with-xml-schema>
                <p:input port="schema">
                    <p:document href="v1schema.xsd"/>
                </p:input>
            </p:validate-with-xml-schema>
        </p:when>

        <p:otherwise>
            <p:validate-with-xml-schema>
                <p:input port="schema">
                    <p:document href="v2schema.xsd"/>
                </p:input>
            </p:validate-with-xml-schema>
        </p:otherwise>
    </p:choose>

    <p:xslt>
        <p:input port="stylesheet">
            <p:document href="stylesheet.xsl"/>
        </p:input>
    </p:xslt>
</p:pipeline>
```

```
<p:pipeline xmlns:p="http://www.w3.org/ns/xproc">
    <p:choose>
        <p:when test="/*[@version < 2.0]">
            <p:validate-with-xml-schema>
                <p:input port="schema">
                    <p:document href="v1schema.xsd"/>
                </p:input>
            </p:validate-with-xml-schema>
        </p:when>

        <p:otherwise>
            <p:validate-with-xml-schema>
                <p:input port="schema">
                    <p:document href="v2schema.xsd"/>
                </p:input>
            </p:validate-with-xml-schema>
        </p:otherwise>
    </p:choose>

    <p:xslt>
        <p:input port="stylesheet">
            <p:document href="stylesheet.xsl"/>
        </p:input>
    </p:xslt>
</p:pipeline>
```

```
<p:pipeline xmlns:p="http://www.w3.org/ns/xproc">
    <p:choose>
        <p:when test="/*[@version < 2.0]">
            <p:validate-with-xml-schema>
                <p:input port="schema">
                    <p:document href="v1schema.xsd"/>
                </p:input>
            </p:validate-with-xml-schema>
        </p:when>

        <p:otherwise>
            <p:validate-with-xml-schema>
                <p:input port="schema">
                    <p:document href="v2schema.xsd"/>
                </p:input>
            </p:validate-with-xml-schema>
        </p:otherwise>
    </p:choose>

    <p:xslt>
        <p:input port="stylesheet">
            <p:document href="stylesheet.xsl"/>
        </p:input>
    </p:xslt>
</p:pipeline>
```

Simplified
(one has to say)

```
<p:pipeline xmlns:p="http://www.w3.org/ns/xproc">
    <p:choose>
        <p:when test="/*[@version < 2.0]">
            <p:validate-with-xml-schema>
                <p:input port="schema">
                    <p:document href="v1schema.xsd"/>
                </p:input>
            </p:validate-with-xml-schema>
        </p:when>

        <p:otherwise>
            <p:validate-with-xml-schema>
                <p:input port="schema">
                    <p:document href="v2schema.xsd"/>
                </p:input>
            </p:validate-with-xml-schema>
        </p:otherwise>
    </p:choose>

    <p:xslt>
        <p:input port="stylesheet">
            <p:document href="stylesheet.xsl"/>
        </p:input>
    </p:xslt>
</p:pipeline>
```

Extended

(one wants to mean)

```
<p:pipeline xmlns:p="http://www.w3.org/ns/xproc" type="foo">
    <p:input port="source" primary="true"/>
    <p:output port="result" primary="true">
        <p:pipe step="xform" port="result"/>
    </p:output>
    <p:choose name="choo1">
        <p:xpath-context>
            <p:pipe step="foo" port="source"/>
        </p:xpath-context>
        <p:when test="/*[@version < 2.0]">
            <p:output port="result">
                <p:pipe step="val1" port="result"/>
            </p:output>
            <p:validate-with-xml-schema name="val1">
                <p:input port="source">
                    <p:pipe step="foo" port="source"/>
                </p:input>
                <p:input port="schema">
                    <p:document href="v1schema.xsd"/>
                </p:input>
            </p:validate-with-xml-schema>
        </p:when>

        <p:otherwise>
            <p:output port="result">
                <p:pipe step="val2" port="result"/>
            </p:output>
            <p:validate-with-xml-schema name="val2">
                <p:input port="source">
                    <p:pipe step="foo" port="source"/>
                </p:input>
                <p:input port="schema">
                    <p:document href="v2schema.xsd"/>
                </p:input>
            </p:validate-with-xml-schema>
        </p:otherwise>
    </p:choose>

    <p:xslt name="xform">
        <p:input port="source">
            <p:pipe step="choo1" port="result"/>
        </p:input>
        <p:input port="stylesheet">
            <p:document href="stylesheet.xsl"/>
        </p:input>
    </p:xslt>
</p:pipeline>
```

Syntax: Verbosity

- Simple pipelines stay simple
  - Defaults take care of all the action
- Complex pipelines require "unleashing the beast", becoming more verbose
  - Explicit contexts, pipes, additional namespaces

Syntax: Namespaces

Feature	URI
Core vocabulary	http://www.w3.org/ns/xproc
Step specific vocabulary	http://www.w3.org/ns/xproc-step
Errors	http://www.w3.org/ns/xproc-error
Standard step library	http://www.w3.org/ns/xproc/1.0

Syntax: Scopes

- Step types must have unique names within the running environment
- Step names must be unique, to be referenced within their environment (i.e., everything in the pipeline)
- Step options are only visible by descendants

Syntax: Binding by URI

```
<p:input port=”source”>
        <p:document href=”http://example.com/input.xml”/>
</p:input>
```

Syntax: Binding by source

```
<p:input port="source">
        <p:pipe step="otherstep" port="result"/>
</p:input>
```

Syntax: Inline binding

```
<p:input port="stylesheet">
        <p:inline>
                <xsl:stylesheet version="1.0">
                        ...
                </xsl:stylesheet>
        </p:inline>
</p:input>
```

Syntax: Empty binding

```
<p:input port=”source”>
        <p:emtpy/>
</p:input>
```

Syntax: Other things

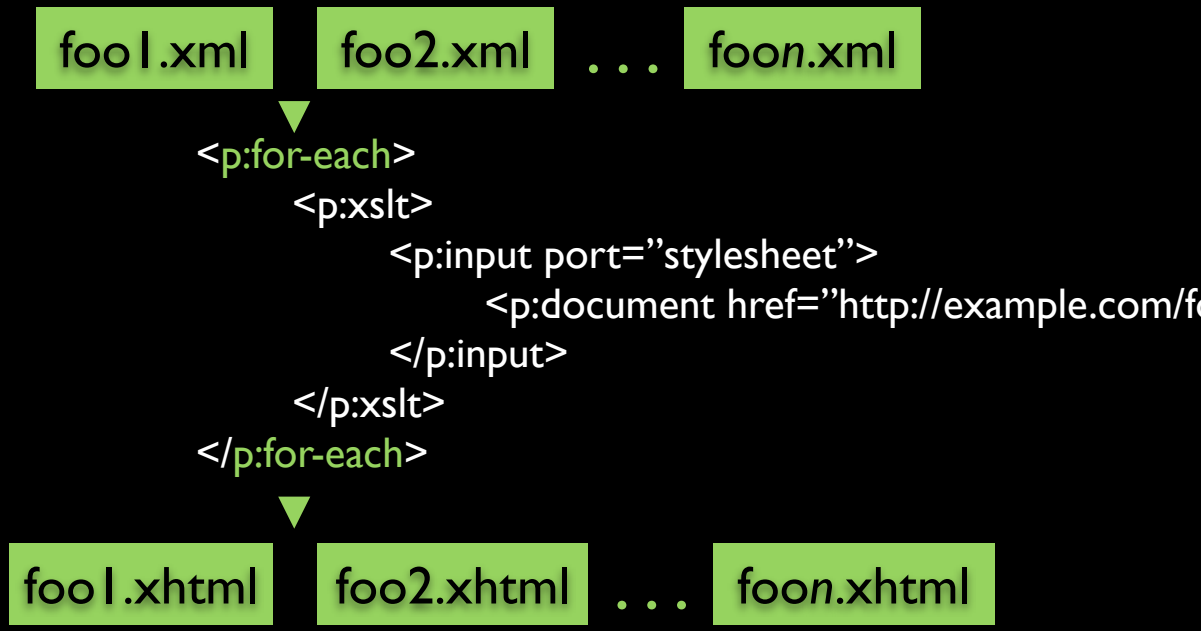
- Globally accepted attributes
  - xml:id, xml:base
- Documentation
  - <p:documentation>
- Extension elements and attributes

XProc: Steps

Steps: p:pipeline

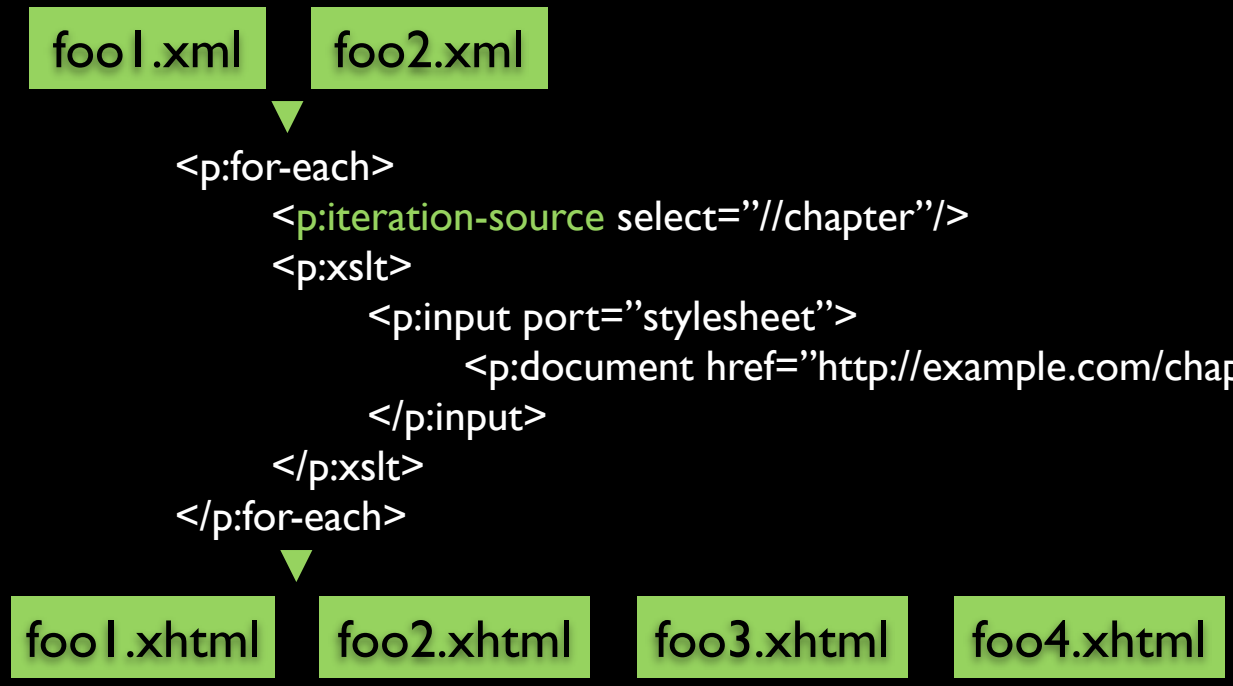
Implicit primary ports: source and result

Steps: p:for-each



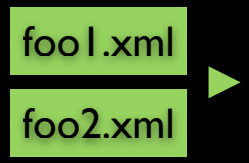
Implicit primary input port: current

Steps: p:for-each



Transform nodes into documents through select

Steps: p:for-each



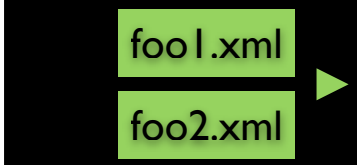

```
<p:for-each name="chapters">
        <p:iteration-source select="//chapter"/>
        <p:output port="html-results">
                <p:pipe step="make-html" port="result"/>
        </p:output>
        <p:output port="fo-results">
                <p:pipe step="make-fo" port="result"/>
        </p:output>

        <p:xslt name="make-html">
                <p:input name="stylesheet">
                        <p:document href="http://example.com/chap2xhtml.xsl"/>
                </p:input>
        </p:xslt>

        <p:xslt name="make-fo">
                <p:input port="source">
                        <p:pipe step="chapters" port="current"/>
                </p:input>
                <p:input port="stylesheet">
                        <p:document href="http://example.com/chap2fo.xsl"/>
                </p:input>
        </p:xslt>
</p:for-each>
```

Supports multiple output bindings

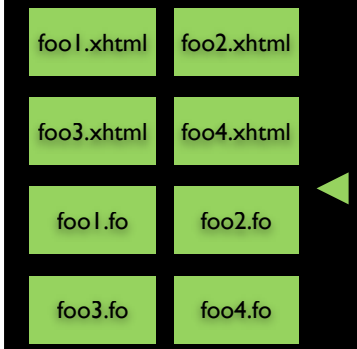
Steps: p:for-each




```
<p:for-each name="chapters">
        <p:iteration-source select="//chapter"/>
        <p:output port="html-results">
                <p:pipe step="make-html" port="result"/>
        </p:output>
        <p:output port="fo-results">
                <p:pipe step="make-fo" port="result"/>
        </p:output>

        <p:xslt name="make-html">
                <p:input name="stylesheet">
                        <p:document href="http://example.com/chap2xhtml.xsl"/>
                </p:input>
        </p:xslt>

        <p:xslt name="make-fo">
                <p:input port="source">
                        <p:pipe step="chapters" port="current"/>
                </p:input>
                <p:input port="stylesheet">
                        <p:document href="http://example.com/chap2fo.xsl"/>
                </p:input>
        </p:xslt>
</p:for-each>
```



Supports multiple output bindings

Steps: p:viewport

```
<p:viewport match="h:div[@class='chapter']" xmlns:h="...">
        <p:insert position="first-child">
                <p:input port="insertion">
                        <p:inline>
                            <hr/>
                        </p:inline>
                </p:input>
        </p:insert>
</p:viewport>
```

Implicit primary input port: current

Steps: p:choose

```
<p:choose>
        <p:when test="/*[@version = 2]">
                <p:validate-with-xml-schema>...</p:validate-with-xml-schema>
        </p:when>

        <p:when test="/*[@version = 1]">
                <p:validate-with-xml-schema>...</p:validate-with-xml-schema>
        </p:when>

        <p:when test="/*[@version]">
                <p:identity/>
        </p:when>

        <p:otherwise>...</p:otherwise>
</p:choose>
```

Steps: p:choose

```
<p:choose>
        <p:xpath-context>...</p:xpath-context>
        <p:when test="/*[@version = 2]">
                <p:validate-with-xml-schema>...</p:validate-with-xml-schema>
        </p:when>

        <p:when test="/*[@version = 1]">
                <p:validate-with-xml-schema>...</p:validate-with-xml-schema>
        </p:when>

        <p:when test="/*[@version]">
                <p:identity/>
        </p:when>

        <p:otherwise>...</p:otherwise>
</p:choose>
```

Expressions default to a common XPath context

Steps: p:choose

```
<p:choose>
    <p:when test="/*[@version = 2]">
        <p:xpath-context>...</p:xpath-context>
        <p:validate-with-xml-schema>...</p:validate-with-xml-schema>
    </p:when>

    <p:when test="/*[@version = 1]">
        <p:xpath-context>...</p:xpath-context>
        <p:validate-with-xml-schema>...</p:validate-with-xml-schema>
    </p:when>

    <p:when test="/*[@version]">
        <p:xpath-context>...</p:xpath-context>
        <p:identity/>
    </p:when>

    <p:otherwise>...</p:otherwise>
</p:choose>
```

But can target multiple XPath contexts

Steps: p:group

```
<p:group>
        <p:option name="db-key" value="something-random-here"/>
        <p:xslt>
                <p:parameter name="key" select="$db-key"/>
                <p:input port="stylesheet">
                        <p:document href="foo.xsl"/>
                </p:input>
        </p:xslt>
</p:group>
```

Encapsulator for local scopes (e.g., p:options)

Steps: p:try

```
<p:try>
        <p:group>
                <p:http-request>...</p:http-request>
        </p:group>

        <p:catch>
                <p:identity>
                        <p:input port=”source”>
                                <p:inline>
                                <html xmlns=”...”>
                                <head><title>error getting resource</head>
                                <body>
                                <h1>error</h1>
                                <h2>getting resource</h2>
                                </body>
                                </html>
                                </p:inline>
                        </p:input>
                </p:identity>
        </p:catch>
</p:try>
```

Prevents and controls errors within pipelines

Steps: p:try

```
<c:errors>
        <c:error type="p:http-request" code="http404">
                <message>Resource not found.</message>
        </c:error>
</c:errors>
```

p:catch default readable port: error

XProc: Other elements

Other elements: Input

Documents

<p:input port="" sequence="" primary=""/> (declare)
<p:input port="" select=""> p:empty, p:pipe, p:document, p:inline </> (use)

Iteration source

<p:iteration-source select=""> p:empty, p:pipe, p:document, p:inline </> (declare)

Viewport source

<p:viewport-source> p:pipe, p:document, p:inline </> (declare)

Other elements: Output

Documents

<p:output port="" sequence="" primary=""/> (declare)
<p:output port="" sequence="" primary=""> p:empty, p:pipe, p:document, p:inline </> (use, only compound steps)

Logging

<p:log port="" href=""/> (declare, can occur in any step)

Serialisation

<p:serialization .../> (huge set of options, declare, can occur just in p:pipeline)

Other elements:
Options and Parameters

Options

<p:option name="" required=""/>
<p:option name="" select=""> p:empty, p:pipe, p:document, p:inline, p:namespaces </>
<p:option name="" value=""> p:namespaces </>
<pfx:stepname option-name="option-value"/>

Parameters

<p:input port="" sequence="" primary="" kind="parameter"/>
<p:parameter name="" select="" port=""> p:empty, p:pipe, p:document, p:inline, p:namespaces </>
<p:parameter name="" value="" port=""> p:namespaces </>

Other elements:
Declaring Steps

```
<p:declare-step type="" psvi-required="" xpath-version="">
        p:input
        p:output
        p:option
        p:import
        p:declare-step
        p:log
        p:serialization
        subpipeline
</p:declare-step>
```

Other elements: Libraries

Library

```
<p:library xpath-version=””>
        p:import
        p:declare-step
</p:library>
```

Import

```
<p:import href=”” />
```

XProc:
Standard Step Library

Standard Step Library:
Required Steps

p:add-attribute, p:add-xml-base, p:compare, p:count, p:delete, p:directory-list, p:error, p:escape-markup, p:http-request, p:identity, p:insert, p:label-elements, p:load, p:make-absolute-uris, p:namespace-rename, p:pack, p:parameters, p:rename, p:replace, p:set-attributes, p:sink, p:split-sequence, p:store, p:unescape-markup, p:string-replace, p:unwrap, p:wrap, p:wrap-sequence, p:xinclude, p:xslt

Standard Step Library:
Optional Steps

p:exec, p:hash, p:uuid, p:validate-with-relax-ng,
p:validate-with-schematron, p:validate-with-xml-schema,
p:www-form-urldecode, p:www-form-urlencode,
p:xquery, p:xsl-formatter

XProc: Recipes

Recipes:
Include, Validate, Transform

```
<p:pipeline xmlns:p="http://www.w3.org/ns/xproc">
    <p:xinclude/>
    <p:validate-with-xml-schema>
        <p:input port="schema">
            <p:document href="schema.xsd"/>
        </p:input>
    </p:validate>
    <p:xslt>
        <p:input port="stylesheet">
            <p:document href="style.xsl"/>
        </p:input>
    </pxslt>
</p:pipeline>
```

Recipes:
Safe processing

```
<p:pipeline xmlns:p="http://www.w3.org/ns/xproc">
        <p:try>
                <p:group>
                        <p:validate-with-xml-schema>
                                <p:input port="schema">
                                        <p:document href="schema.xsd"/>
                                </p:input>
                        </p:validate-with-xml-schema>
                        <p:xslt>
                                <p:input port="stylesheet">
                                        <p:document href="style.xsl"/>
                                </p:input>
                        </pxslt>
                </p:group>
                <p:catch>
                        <p:output port="result">
                                <p:inline>
                                        <html>...</html>
                                </p:inline>
                        </p:output>
                </p:catch>
        </p:try>
</p:pipeline>
```

Recipes:
Removal

Elements (including subtrees)

```
<p:pipeline xmlns:p="http://www.w3.org/ns/xproc">
    <p:delete match="chapter[@draft='yes']" />
</p:pipeline>
```

Elements (excluding subtrees)

```
<p:pipeline xmlns:p="http://www.w3.org/ns/xproc">
    <p:unwrap match="bad-wrapper" />
</p:pipeline>
```

Attributes

```
<p:pipeline xmlns:p="http://www.w3.org/ns/xproc">
    <p:delete match="@draft" />
</p:pipeline>
```

Recipes:
Transform set of files

```
<p:pipeline xmlns:p="http://www.w3.org/ns/xproc" xmlns:c="...">
    <p:directory-list path="." include-filter=".*docx$"/>
    <p:for-each>
        <p:iteration-source select="//c:file"/>
        <p:load>
            <p:option name="href" select="@name"/>
        </p:load>
        <p:xslt>
            <p:input port="stylesheet">
                <p:document href="docx2odf.xsl"/>
            </p:input>
        </p:xslt>
    </p:for-each>
</p:pipeline>
```

Thank you.

Working Group Homepage
http://www.w3.org/XML/Processing/

Current state of the spec
http://www.w3.org/XML/XProc/docs/langspec.html

Official URL for the spec
http://www.w3.org/TR/xproc/

Known implementations (not up-to-date with spec)
http://del.icio.us/ndw/xprocimpl