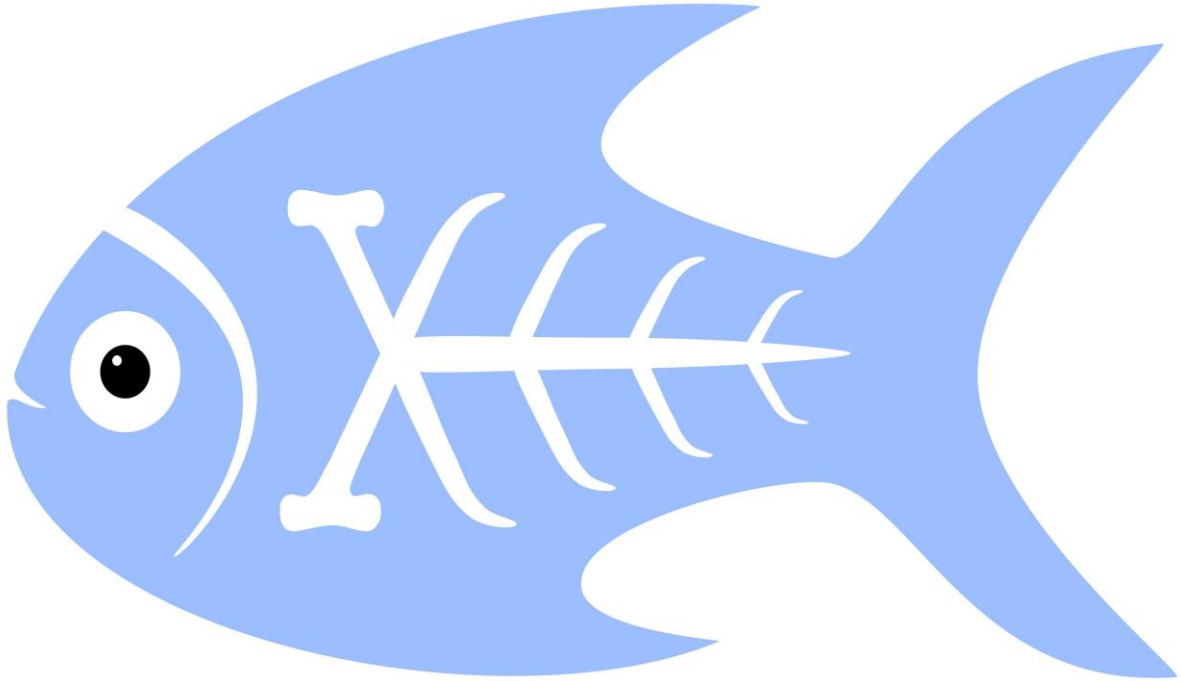
Introduction to XProc 3.0 – Part 1

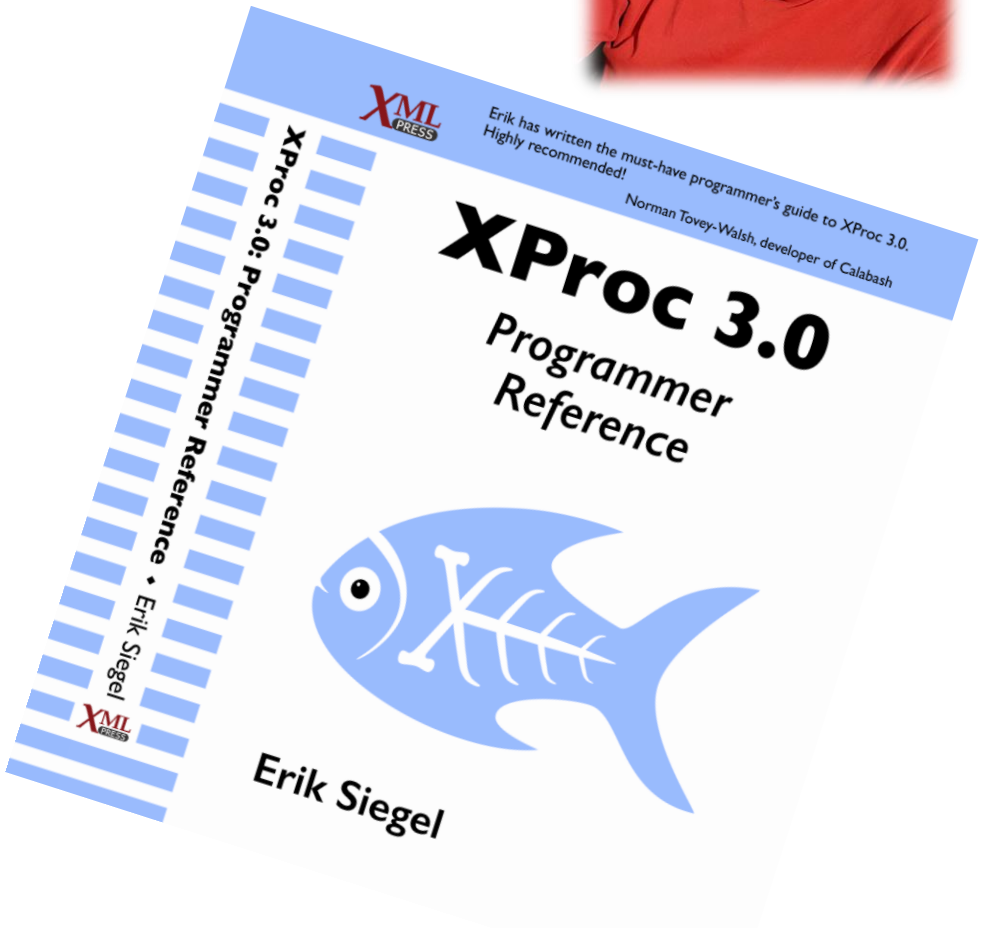
Markup UK 2020
Webinar



Who Am I?

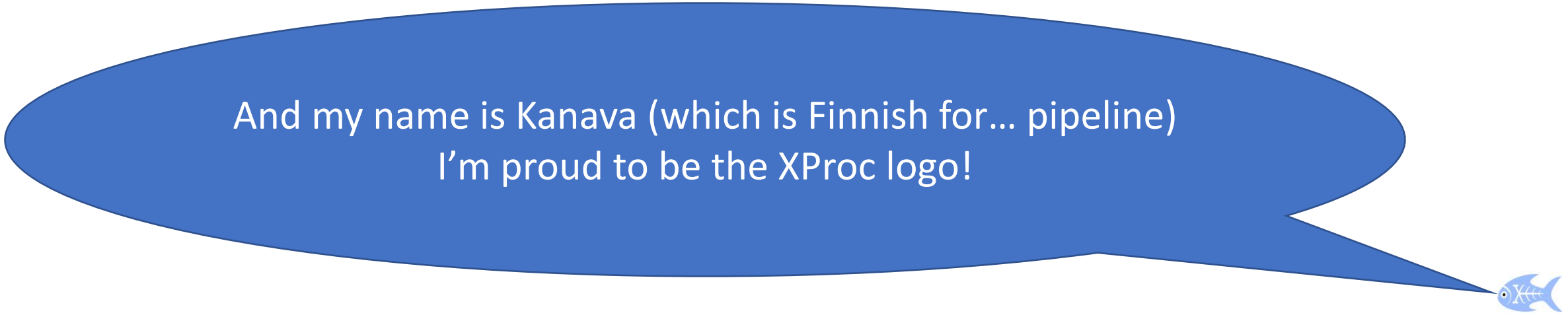
- Erik Siegel
- Content Engineer, XML Specialist, Technical Writer
- Company: Xatapult
  - Groningen, The Netherlands
  - Customers mostly in publishing and standardization
- Part of the XProc 3.0 editing committee
- Writer of the XProc 3.0 Programmer Reference
- Contact:

erik@xatapult.nl
www.xatapult.com
www.linkedin.com/in/esiegel/
+31 6 53260792

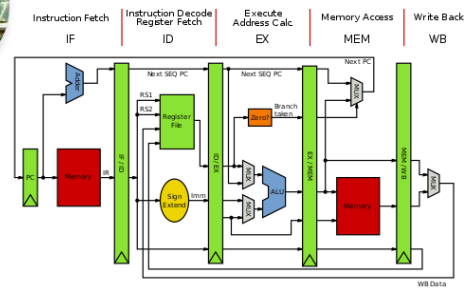


XProc?

- XProc is an XML based programming language for complex data processing - pipelining
- Extensible set of small, sharp tools for creating and transforming XML and other documents
- V1.0 available (two processor implementations to run your pipelines)
- Specification and implementation V3.0 under development



Why should I bother?



- Pipelines are ubiquitous all around us
- Solve problems with a set of small, sharp tools that combine in many ways
  - Like the UNIX command line
- Very natural choice for document processing
- Compose small tools into something bigger, pipelines…
- XProc beats the alternatives

A successful example of large-scale application of XProc (1.0) pipelines doing document engineering:
https://www.le-tex.de/en/transpect.html

Important links

- **XProc 3.0:**
  - **Specification: http://spec.xproc.org**
  - Github: https://github.com/xproc/
  - W3C: https://www.w3.org/community/xproc-next/
- Morgana XProc processor: https://www.xml-project.com/
- This webinar: https://github.com/xatapult/markupuk-2020
- There are some introductory articles on https://www.xml.com/

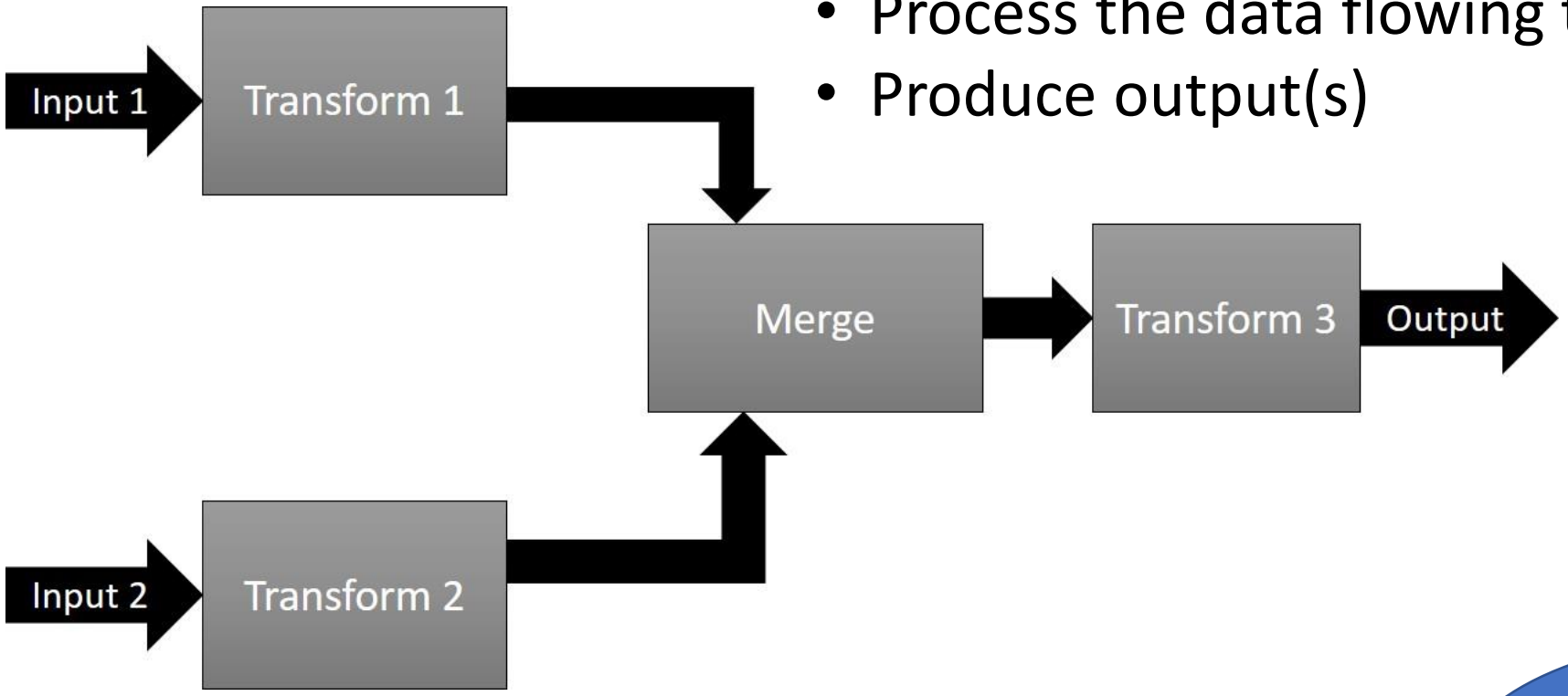
- **XProc 1.0:**
  - **Specification: https://www.w3.org/TR/xproc/**
  - **XML Calabash processor: https://xmlcalabash.com/**
  - **Morgana XProc processor: https://www.xml-project.com/**

Running XProc 3.0 examples yourself

- Download Morgana by following the download link: https://www.xml-project.com/
- Unzip the zip
- Add the main Morgana directory to your system's path
- Switch (`cd`) to the directory with the pipeline you want to run (assume this is called `pipeline.xpl`)
- View the command line options:
  - `Morgana`
- Run with no specific input:
  - `Morgana pipeline.xpl`
- … and with a specific input file for the source port:
  - `Morgana pipeline.xpl –input source:input.xml`
- … and write the result port's output to a file:
  - `Morgana pipeline.xpl –input:source:input.xml –output:result=output.xml`

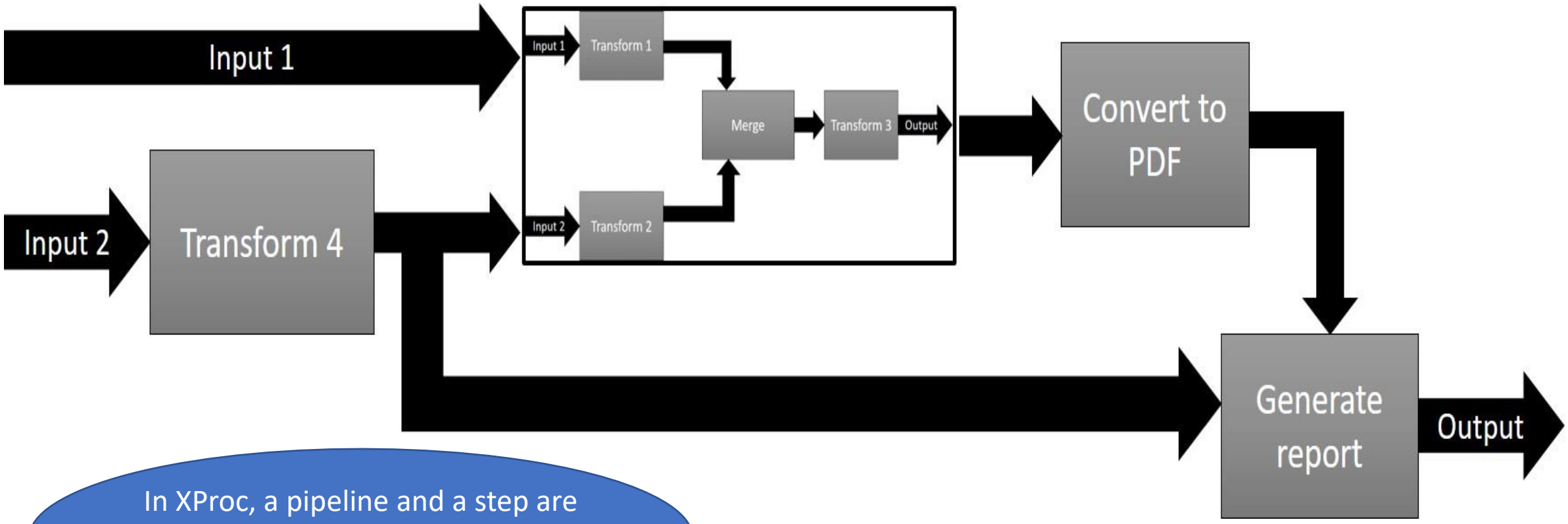
Pipelines, steps

- Document(s) as input
- Process the data flowing through using steps
- Produce output(s)



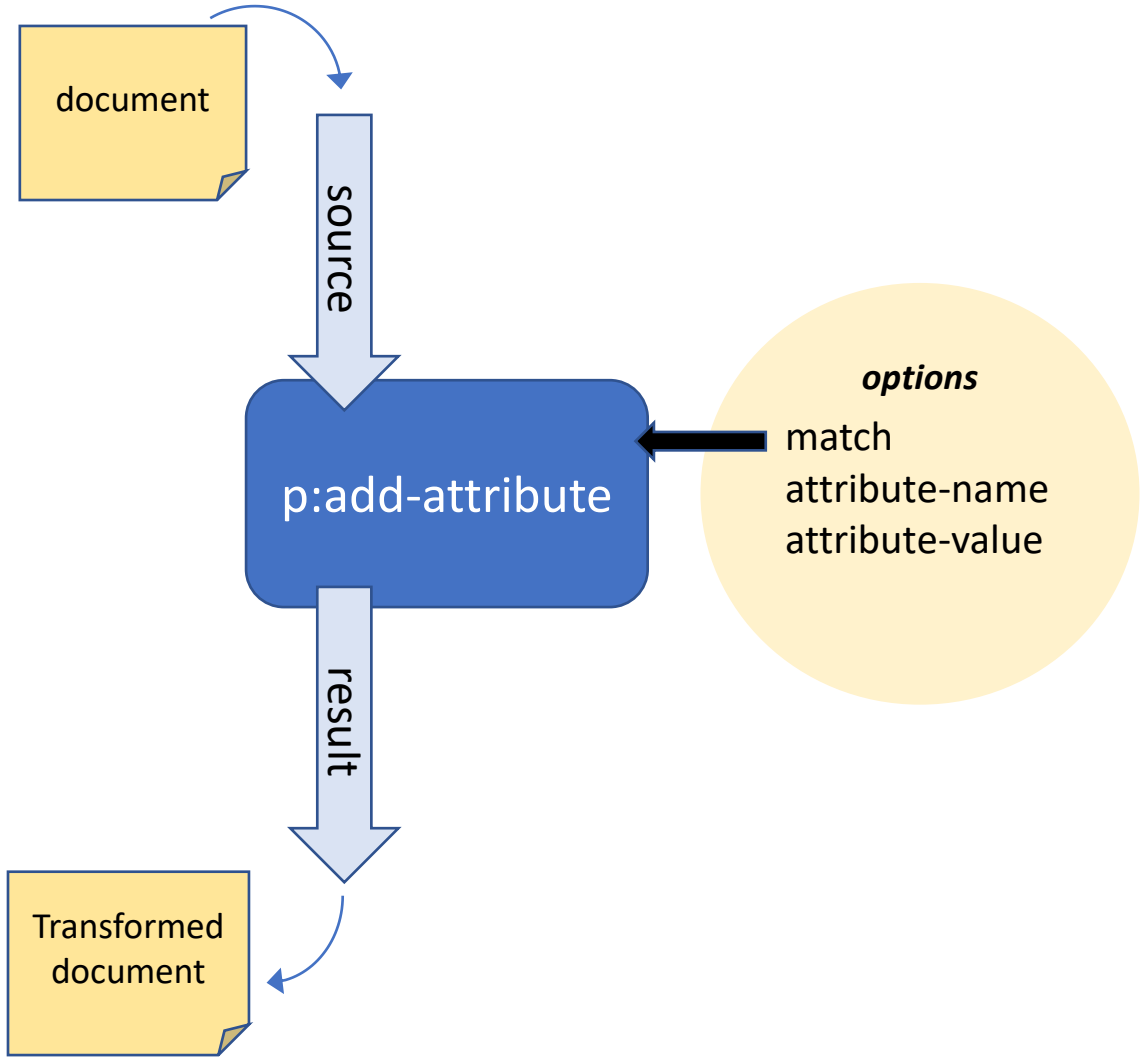
Documents can be of any type, not just XML!

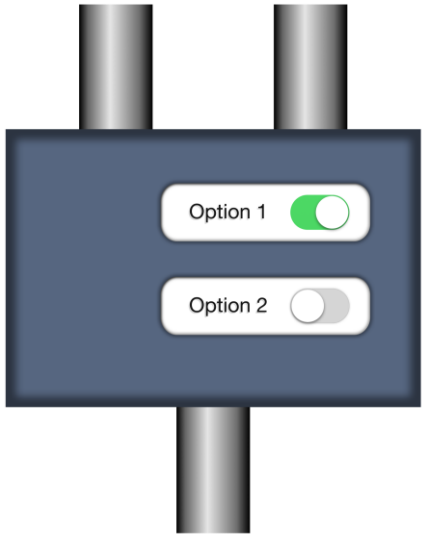
Pipelines, steps



In XProc, a pipeline and a step are essentially the same. The terms can be used interchangeably!

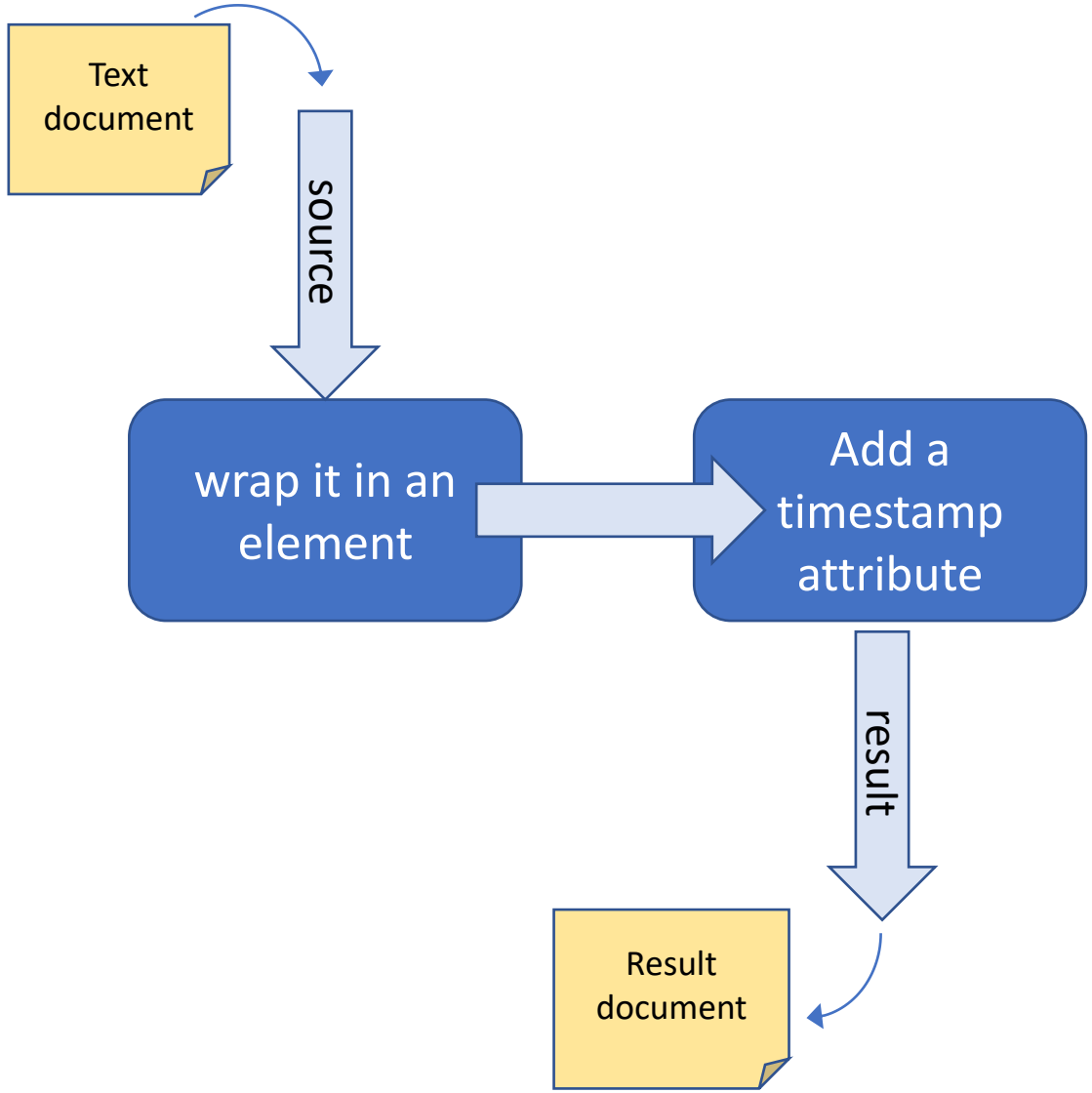
Steps/pipelines, ports, options



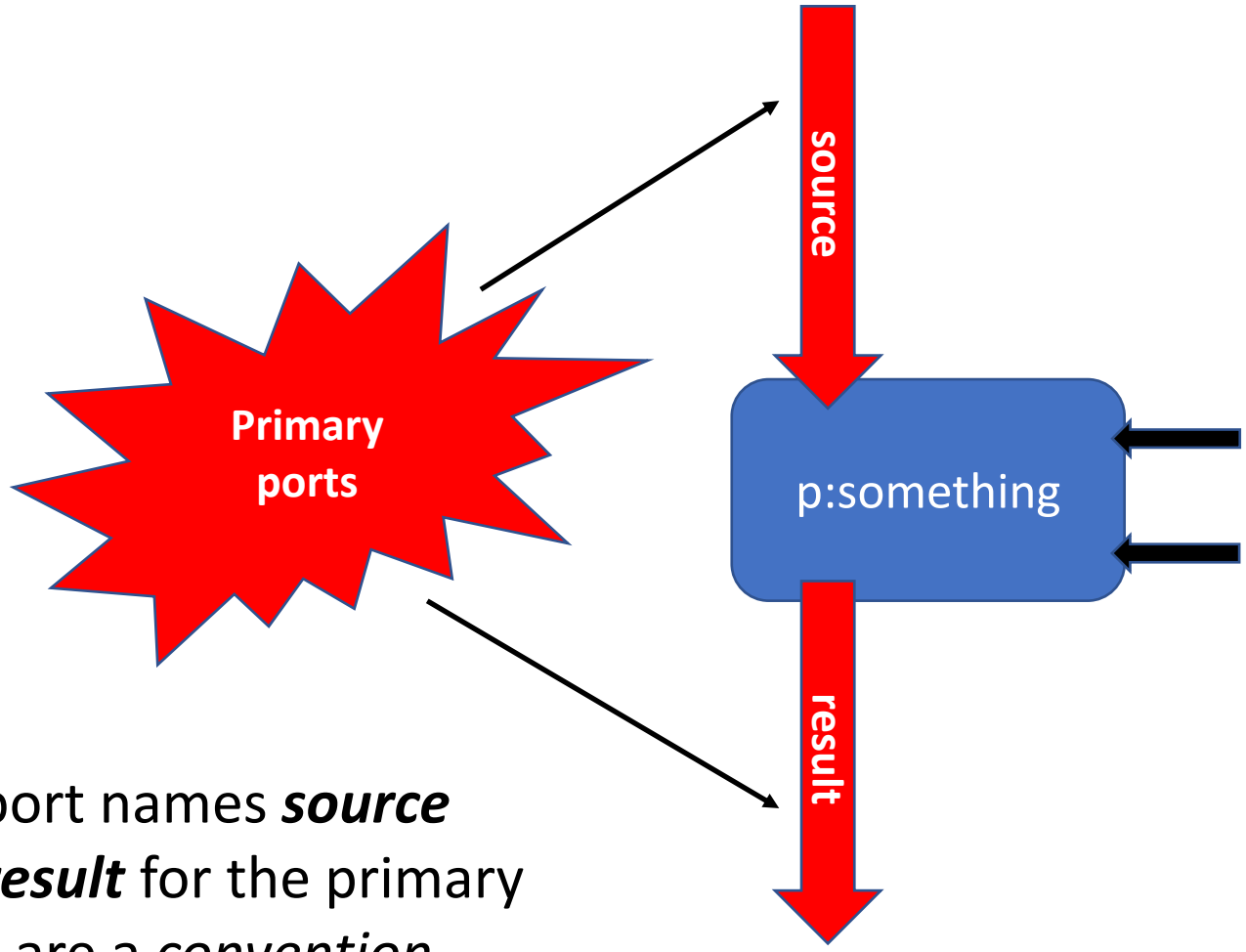


Have a look ate the step specification:
http://spec.xproc.org/master/head/steps/#c.add-attribute

Example 1: `markupuk-2020/101-A/example-1/example-1.xpl`



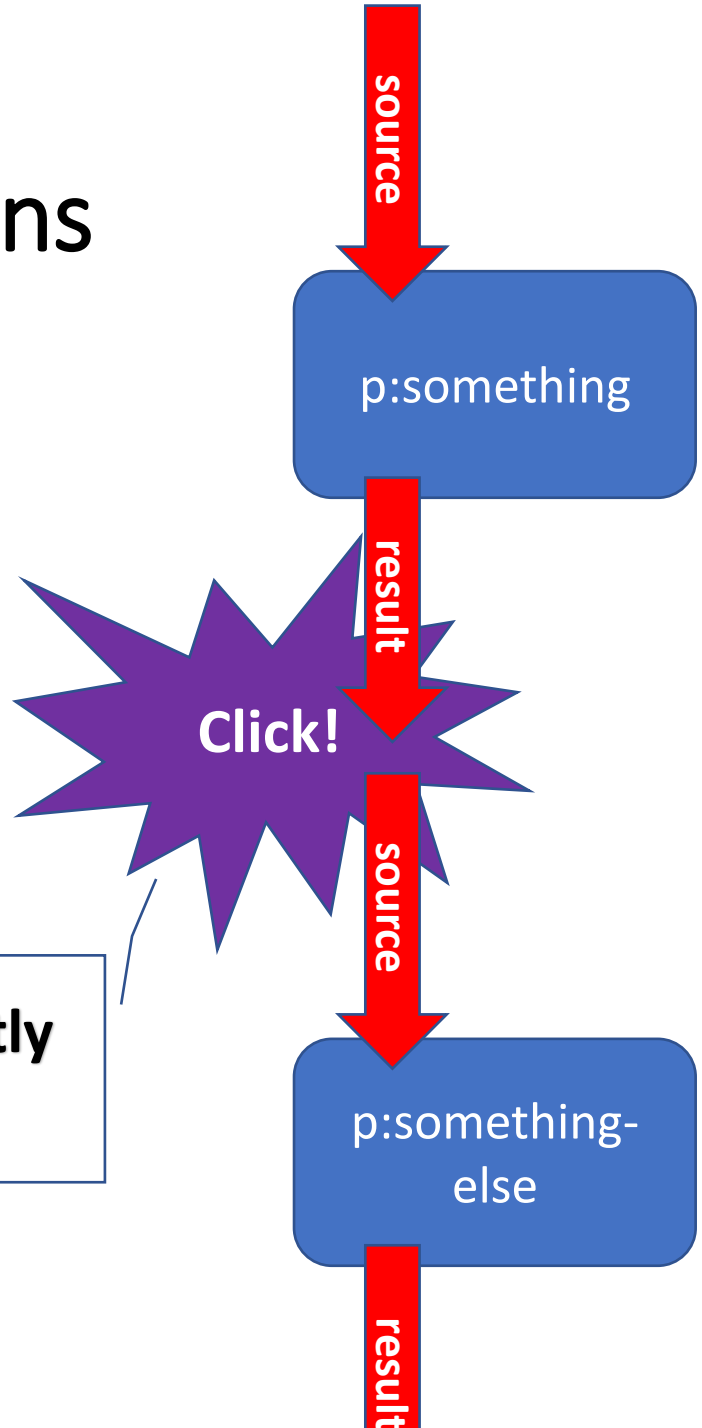
Primary ports



The port names ***source*** and ***result*** for the primary ports are a *convention*

Not all ports are created equal…

Primary ports, implicit connections



Primary ports implicitly connect

Think of primary ports as having little *magnets* that *snap* automagically together

Primary ports, implicit connections

If a step has only a single input or output port, they're primary by default. But you can set the primary status *explicitly* using a primary="true/false" attribute here.

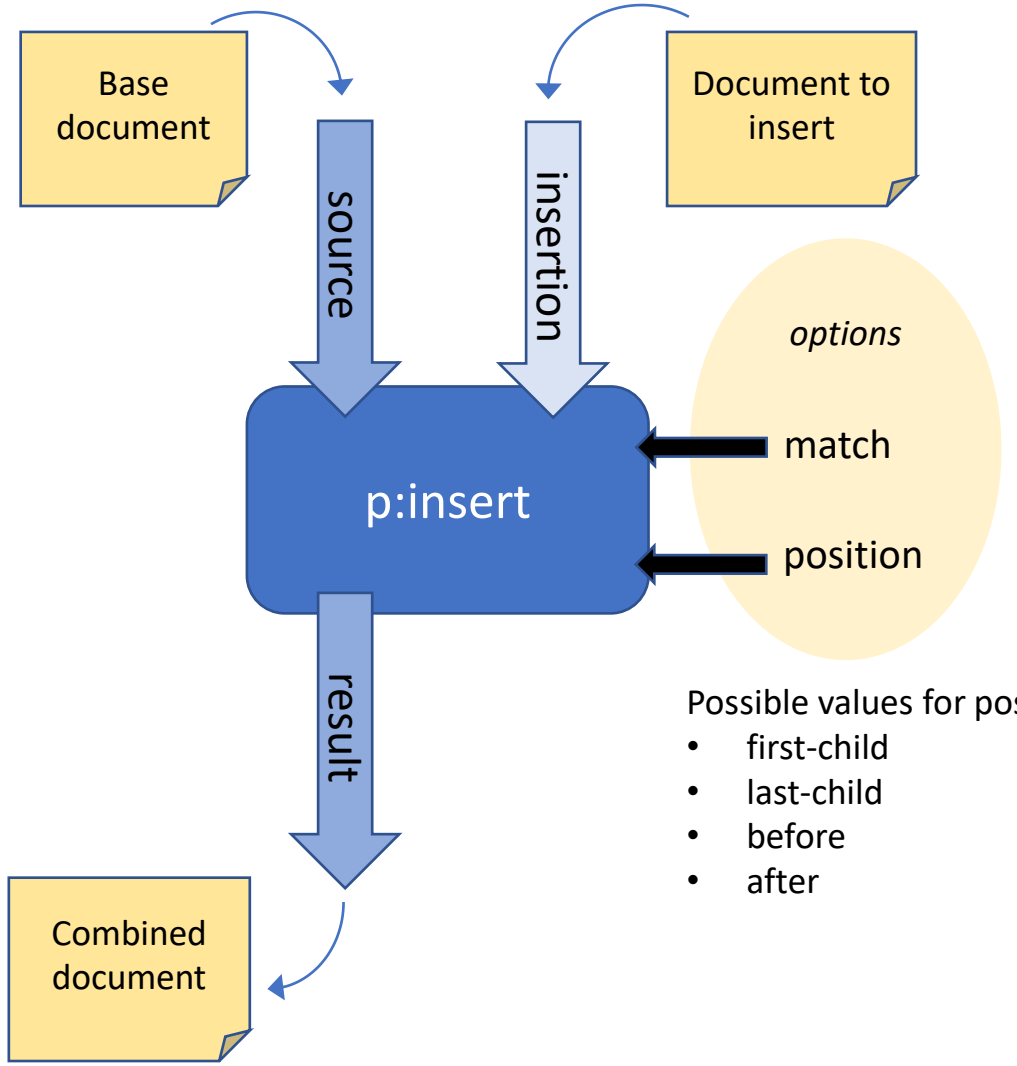
Implicit connection of primary input port to first step

Implicit connection of last step to primary output port

Implicit connection of steps

```
<p:declare-step … >

  <p:input port="source"/>
  <p:output port="result"/>

  <p:wrap-sequence … />

  <p:add-attribute … />

</p:declare-step>
```

The p:insert step



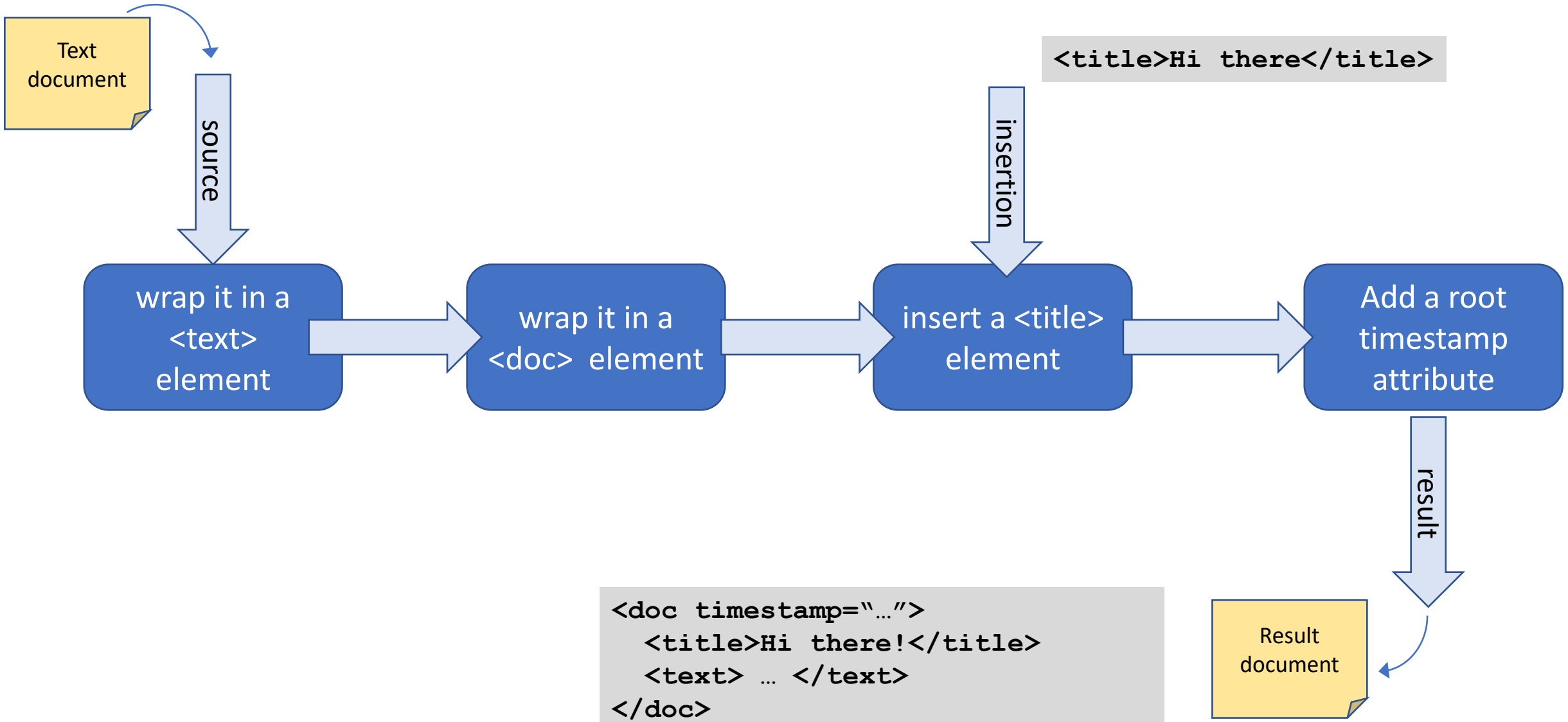
Possible values for position:

- first-child
- last-child
- before
- after

See:
http://spec.xproc.org/master/head/steps/#c.insert

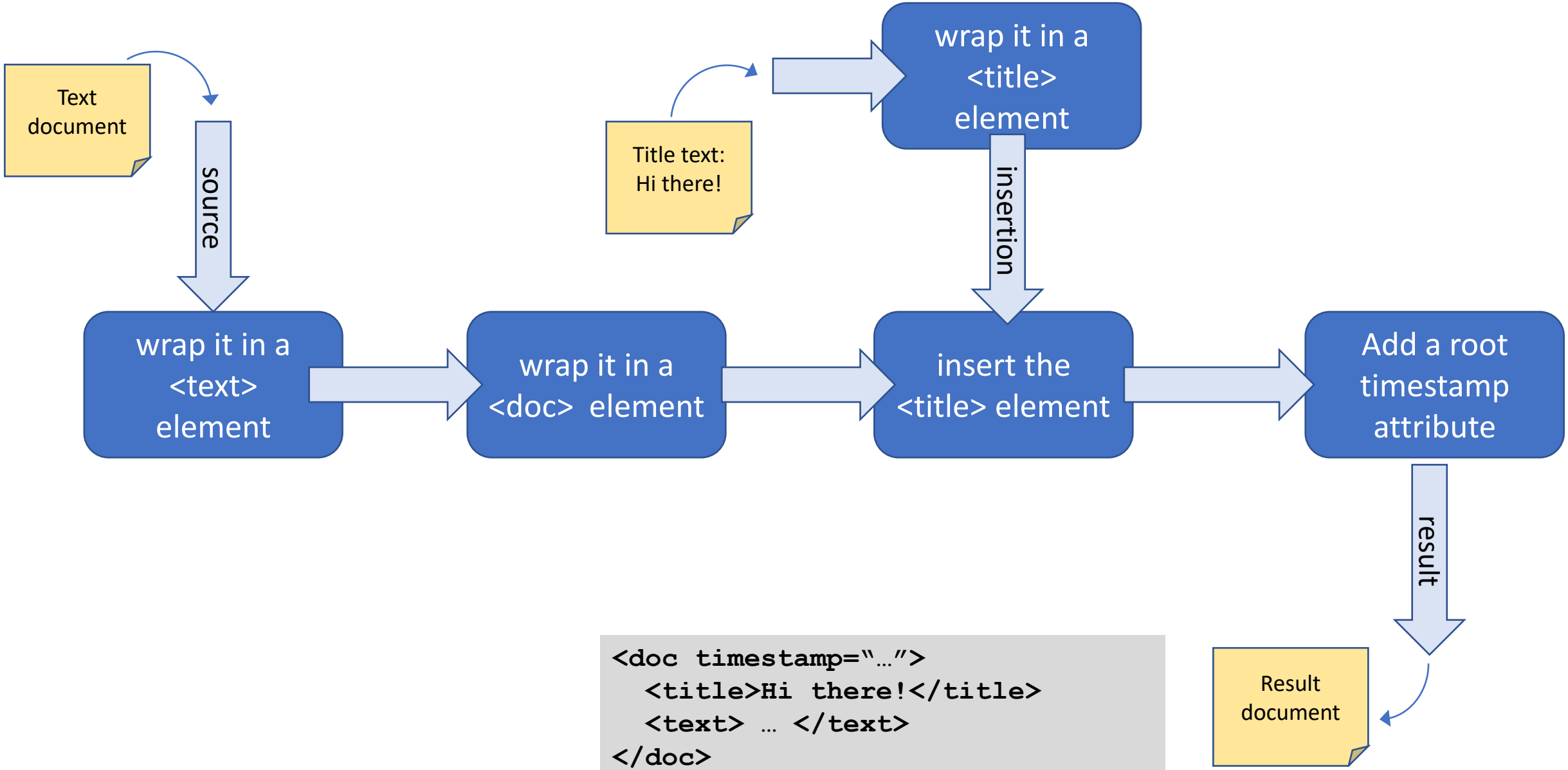
The source and result ports are primary, the insertion port is not…

Example 2: `markupuk-2020/101-A/example-2/example-2a.xpl`



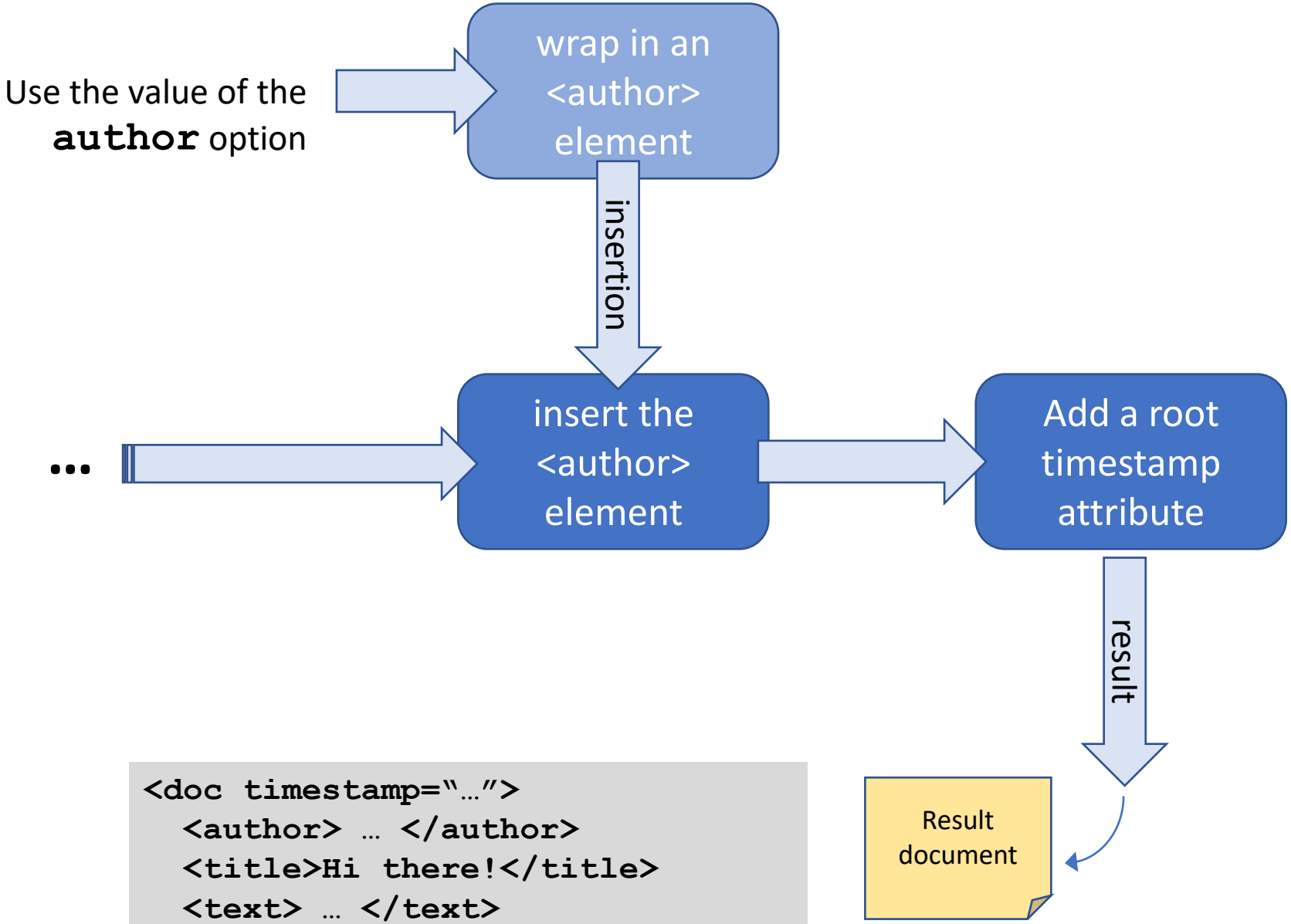

```
<doc timestamp="…">
  <title>Hi there!</title>
  <text> … </text>
</doc>
```

Example 4: `markupuk-2020/101-A/example-4/example-4a.xpl`




```
<doc timestamp="…">
  <title>Hi there!</title>
  <text> … </text>
</doc>
```

Example 5: `markupuk-2020/101-A/example-5/example-5.xpl`




```
<doc timestamp="…">
  <author> … </author>
  <title>Hi there!</title>
  <text> … </text>
</doc>
```

Wrap up:

- XProc is a *pipeline* language for documents, it chains *steps*
- Documents flow in and out of steps through *ports*
- One input and one output port can be *primary*: These ports automatically connect
  - Primary ports are called `source` and `result` by convention
- You can connect a port to:
  - Another port (either *implicit* for primary ports or *explicit*: `<p:pipe>` or `@pipe`)
  - To a document stated inline (`<p:inline>`)
  - To a document on disk (`<p:document>` or `@href`)
- *Options* are additional switches for the steps and/or your pipelines

Goodbye and thank the fish!

Your guide today: Erik Siegel – erik@xatapult.nl

Specification: https://spec.xproc.org/

Processors:

- Morgana: https://www.xml-project.com/
- Calabash: https://xmlcalabash.com/

Articles on XProc: https://www.xml.com

Book: https://xmlpress.net/publications/xproc-3-0/