DocEng 2021 — Limerick, Ireland 2021-08-25

Small-step pipelines reduce the complexity of XSLT / XPath programs

https://gioele.io/p/doceng2021

Marcel Schaeben
m.schaeben@uni-koeln.de
Cologne Center for eHumanities
Köln, Germany

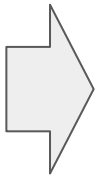
Gioele Barabucci
gioele.barabucci@ntnu.no
Norwegian University of Science and Technology
Trondheim, Norway

Problem

Programs grow as data gets more complex (1)

Input data

```
<expenses>
    <!-- all expenses in EUR -->
    <expense id="1" person="gb"
value="10 "/>
    <expense id="2" person="ms"
value="1000" />
    [...]
```



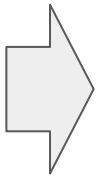
Business logic

```
<xsl:value-of
    select="sum(//expense/@value)"
/>
```

Programs grow as data gets more complex (2)

Input data

```
<expenses>
    <expense id="1" person="gb"
value="10" />
    <expense id="2" person="gb"
value="1000" />
    <expense id="3" person="ms"
value="30" currency="usd" />
    <expense id="4" person="gb"
value="350" currency="Eur"/>
```



Business logic

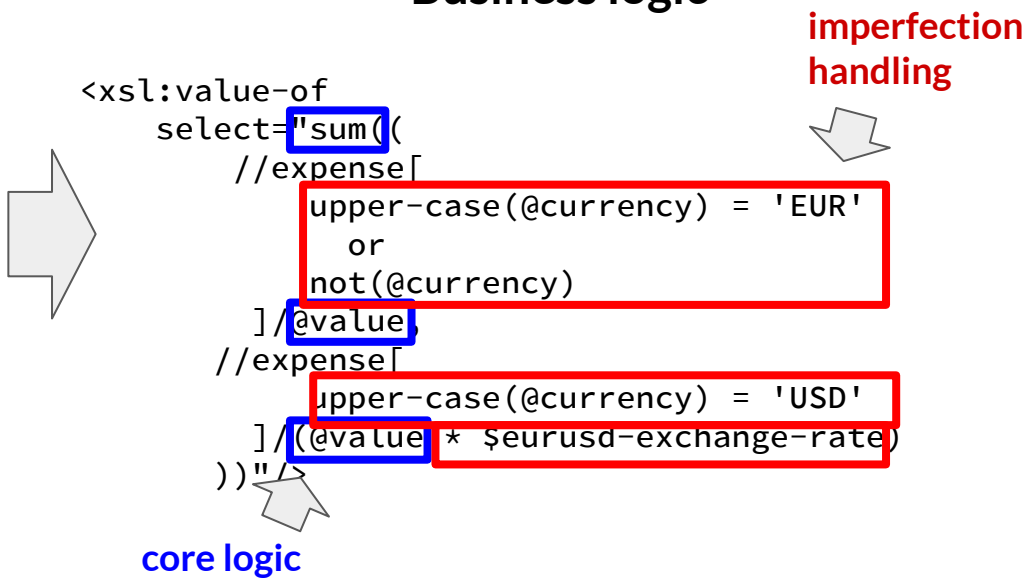
```
<xsl:value-of
    select="sum((
        //expense[
            upper-case(@currency) = 'EUR' or
            not(@currency)
            ]/@value,
        //expense[
            upper-case(@currency) = 'USD'
            ]/(@value * 0.9)
    ))"/>
```

More imperfection handling than core logic

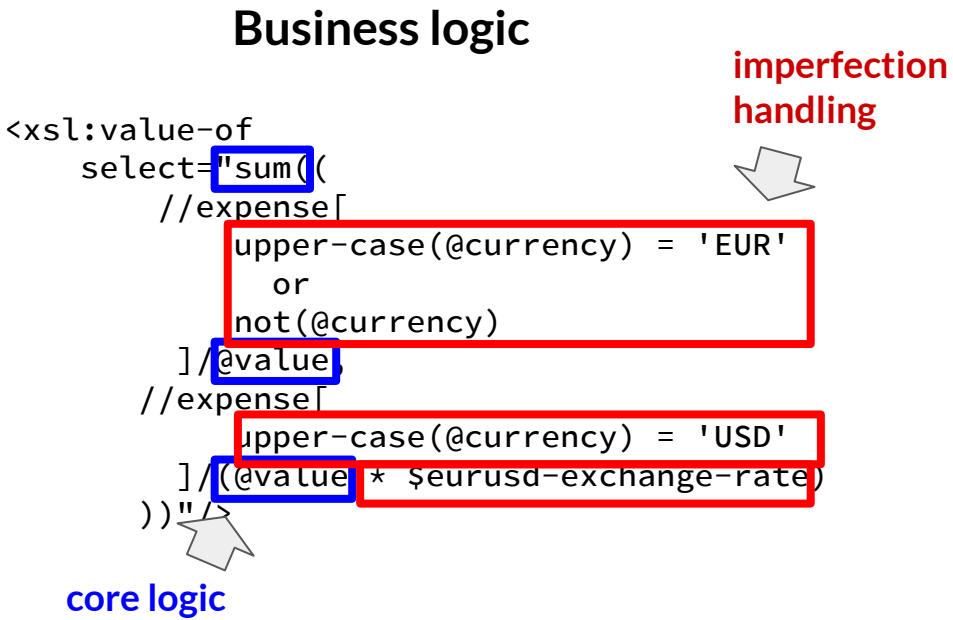
Input data

```
<expenses>
    <expense id="1" person="gb"
value="10" />
    <expense id="2" person="gb"
value="1000" />
    <expense id="3" person="ms"
value="30" currency="usd" />
    <expense id="4" person="gb"
value="350" currency="Eur"/>
```

Business logic



Harder to work with



1. Less readable
2. Harder to debug
3. Harder to maintain
4. Harder to extend with new functionalities

Solution? First fix, then compute

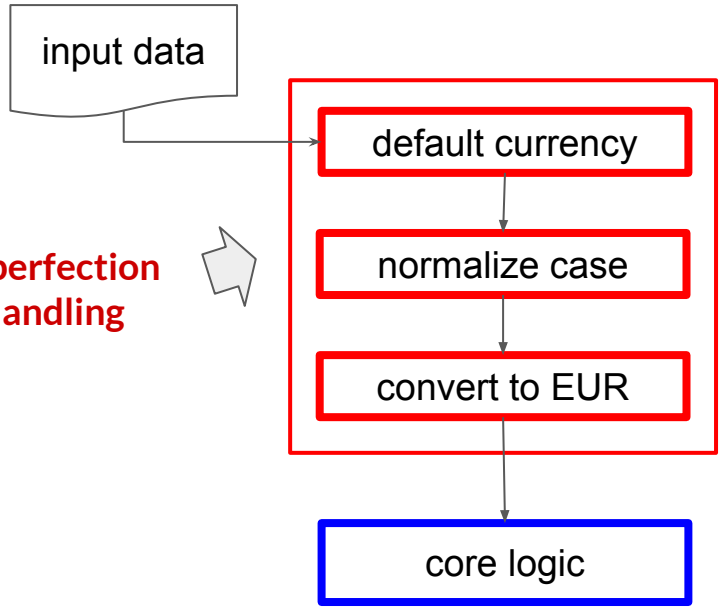
Conventional

```
<xsl:value-of
    select="sum((
        //expense[
            upper-case(@currency) = 'EUR'
              or
            not(@currency)
          ]/@value,
        //expense[
            upper-case(@currency) = 'USD'
          ]/(@value * $eurusd-exchange-rate)
        ))"/>
```

core logic

imperfection handling

Pipeline

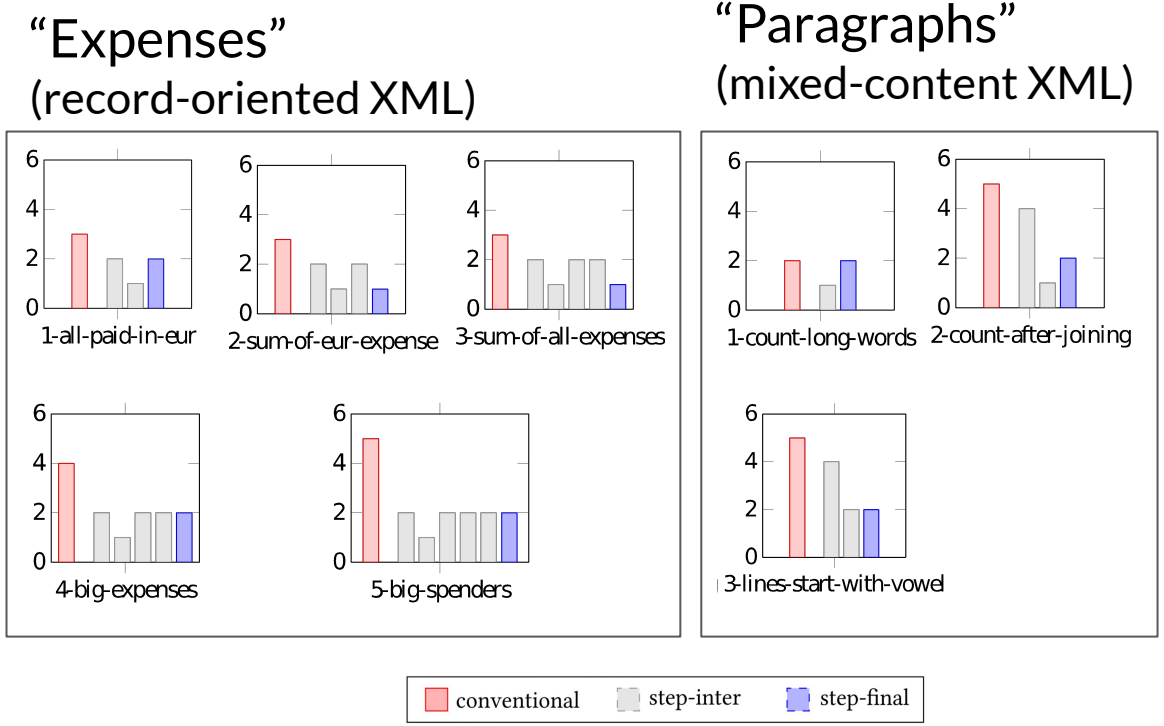


Results

Results (in a nutshell)

After rewriting a monolithic program as small-step pipeline…

- Core task up to 2.5x **less complex** than conventional program.
- Peak complexity of pipeline is always lower or equal than that of conventional program.



Small-step pipelines

Small-step pipelines

1. **Curation before analysis**
   Data is curated (i.e., imperfections are smoothed out, edge-cases are handled) before it is analyzed.
2. **Small steps**
   The data-curation phase is deconstructed into small steps.
3. **One problem at a time**
   Each data-curation step fixes only one specific problem.
4. **Either read or fix**
   A data-curation step can only use a piece of data or fix it, but not both.

Small-step pipelines: “Either use or fix”

Use and fix at the same time (WRONG)

```
total = (attr(“previous”) ? @previous : DEFAULT_PREVIOUS) + @current
```

First fix, then use (BETTER)

Step 1:

```
if !attr(“previous”) {
    @previous = DEFAULT_PREVIOUS
}
```

Step 2:

```
total = @previous + @value
```

Projects that have adopted small-step pipelines (selected)

- **Cologne Sanskrit Lexikon / LAZARUS**
  https://cceh.uni-koeln.de/portfolio/lazarus/
  University of Cologne, CCeH, 2013
- **HallerNet https://hallernet.org/**
  Albrecht-Haller-Stiftung, Universität Bern, CCeH, 2016
- **The School of Salamanca**
  Akademie der Wissenschaften und der Literatur Mainz, 2018
- **Marco Polo ENGH**
  University Ca' Foscari Venice, 2021

Prove better?

Methodology: comparing complexity

monolith version

task_1()

task_2()

task_3()

pipeline version

t1_step_1() *t1_step_2()* *t1_step_3()*

t2_step_1() *t2_step_2()*

t3_step_1() *t3_step_2()* *t3_step_3()* *t3_step_4()*

→ *which is better?*

What do we mean by “better”?

Same outcome, but…

- more readable
- more understandable
- easier to maintain
- less prone to errors

What do we mean by “better”?

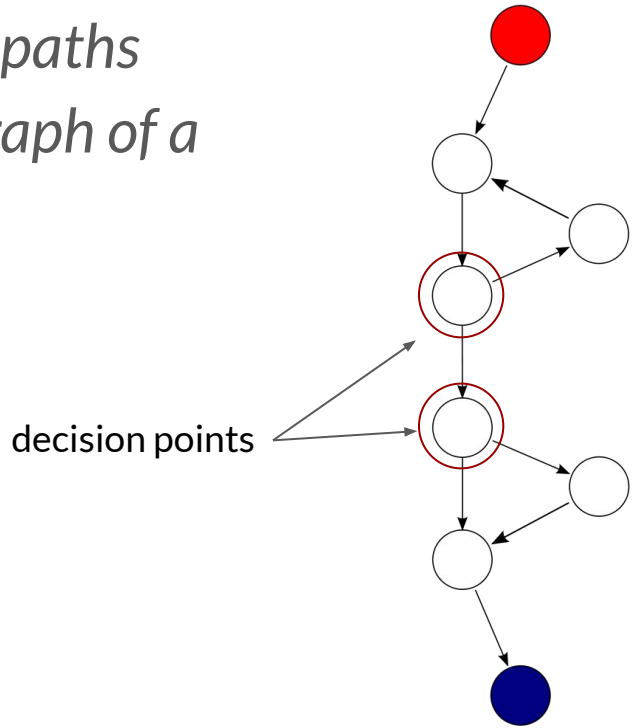
Same outcome, but…

- more readable
- more understandable
- easier to maintain
- less prone to errors

→ McCabe cyclomatic complexity

McCabe cyclomatic complexity (McCabe 1976)

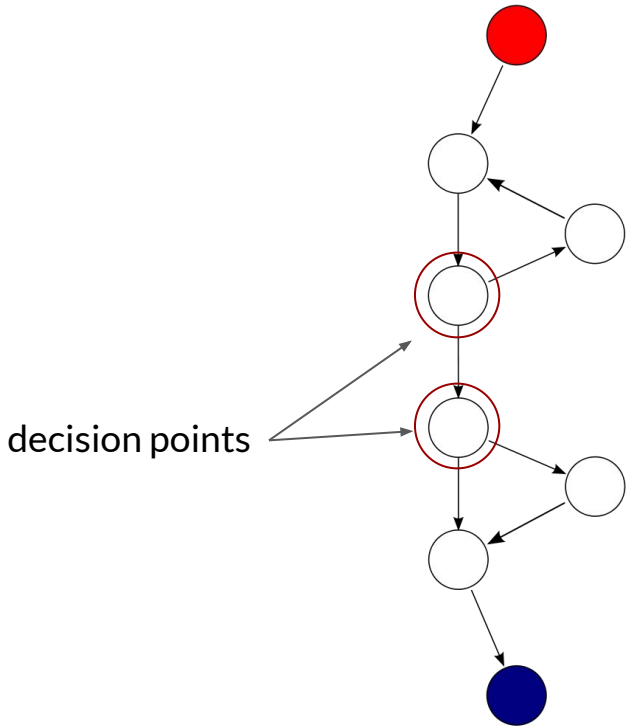
Number of independent paths through the execution graph of a program



Source:
https://en.wikipedia.org/wiki/Cyclomatic_complexity

What increases complexity?

- if conditions
- boolean expressions
- for/while loops
- switch cases
- ...



Source:
https://en.wikipedia.org/wiki/Cyclomatic_complexity

Methodology: comparing complexity

monolith version

task_1()

task_2()

task_3()

pipeline version

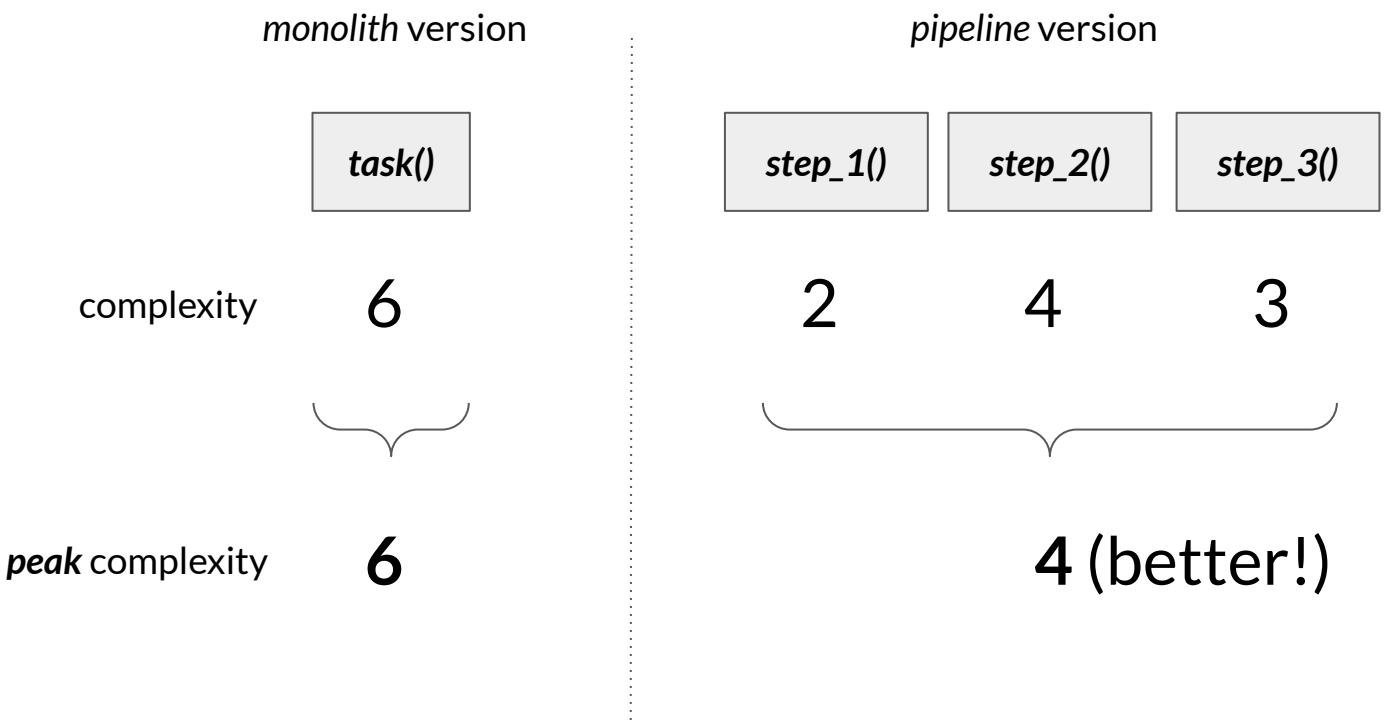
t1_step_1() *t1_step_2()* *t1_step_3()*

t2_step_1() *t2_step_2()*

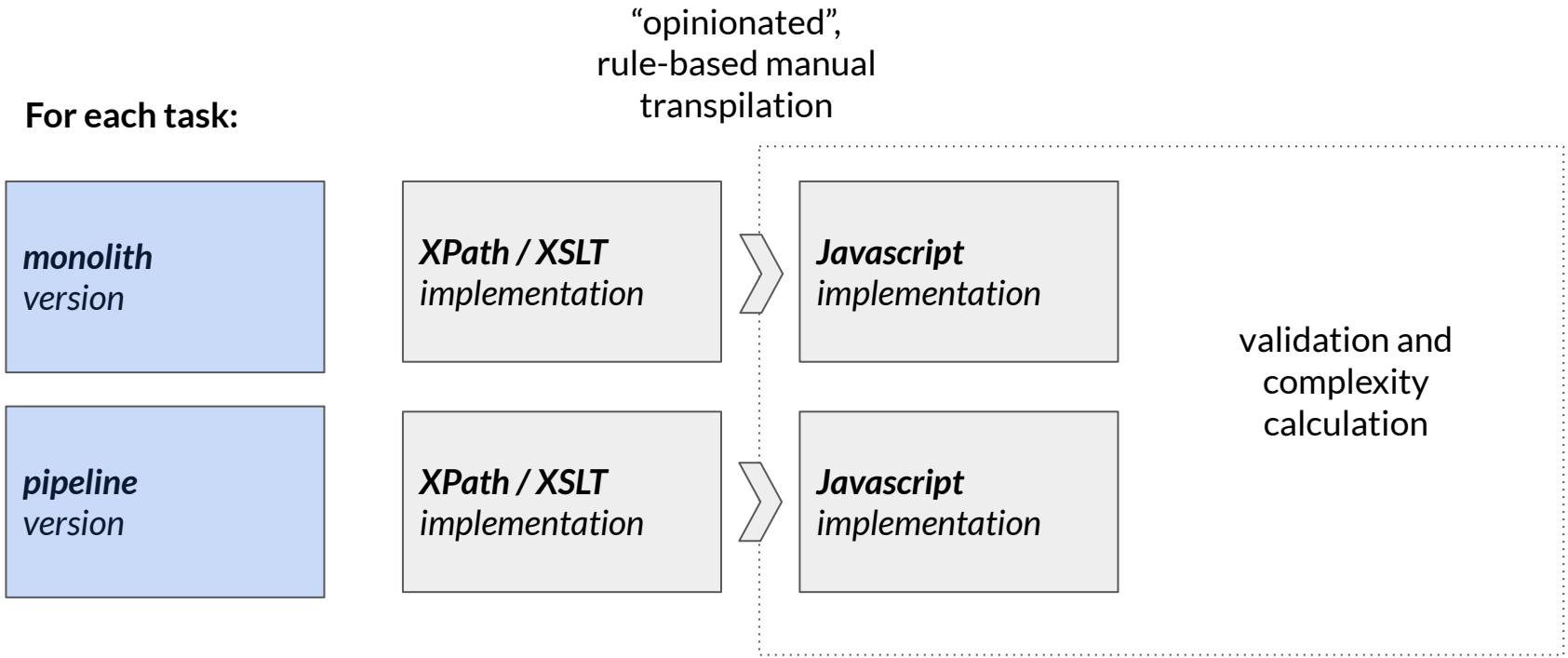
t3_step_1() *t3_step_2()* *t3_step_3()* *t3_step_4()*

→ *which is better?*

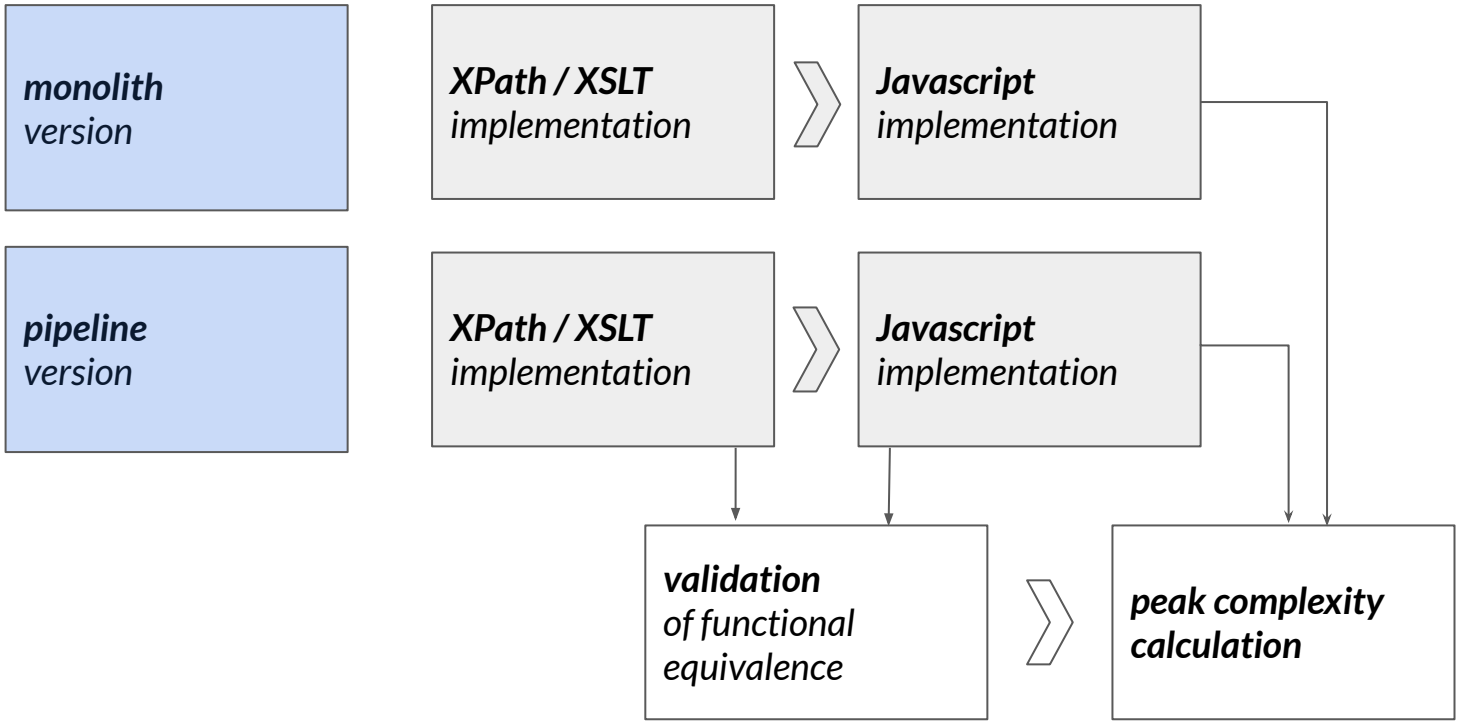
Peak cyclomatic complexity



Calculating complexity



Calculating complexity



Reproducible methodology

Dataset

Original (XSTL/XPath) and transpiled (JS) programs

Validation scripts

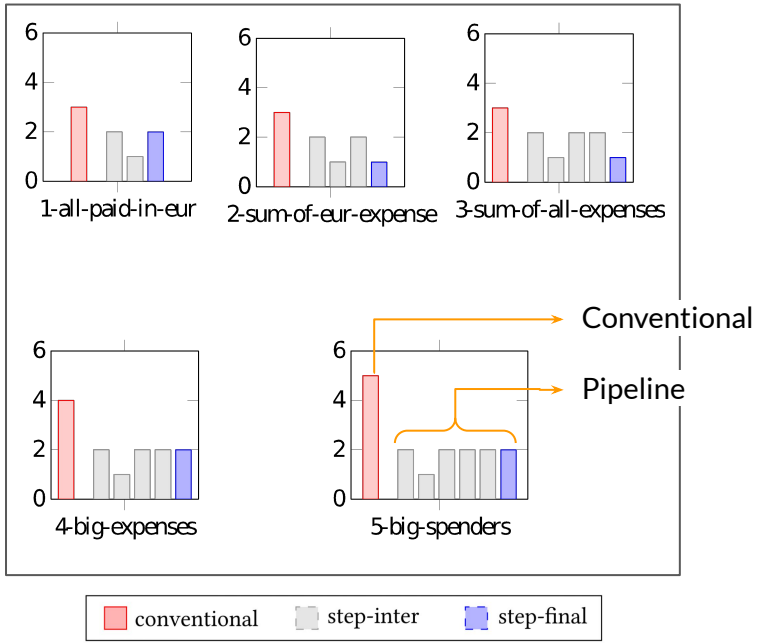
Complexity analysis scripts

https://zenodo.org/record/5115788

Results

Result 1: big complexity reduction when data is complex

"Expenses"
(record-oriented XML)



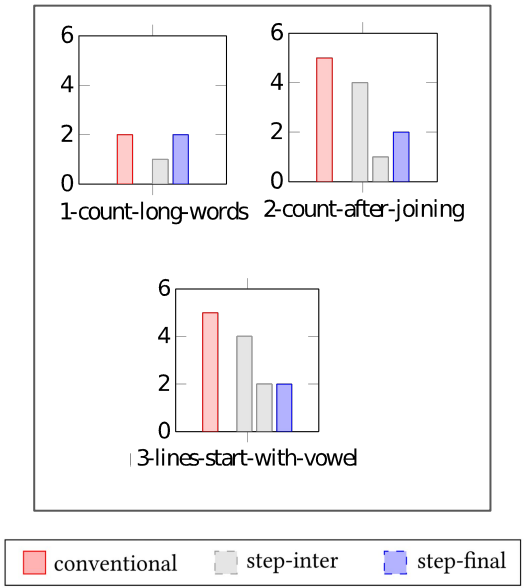
Task	Peak complexity		C/SSP
	C	SSP	
1-all-paid-in-eur	3	2	1.5x more complex
2-sum-of-eur-expenses	3	2	1.5x more complex
3-sum-of-all-expenses	3	2	1.5x more complex
4-big-expenses	4	2	2x more complex
5-big-spenders	5	2	2.5x more complex

Conventional

Simple-step pipeline

Result 2: incremental complexity reduction → lean core

"Paragraphs"
(mixed-content XML)



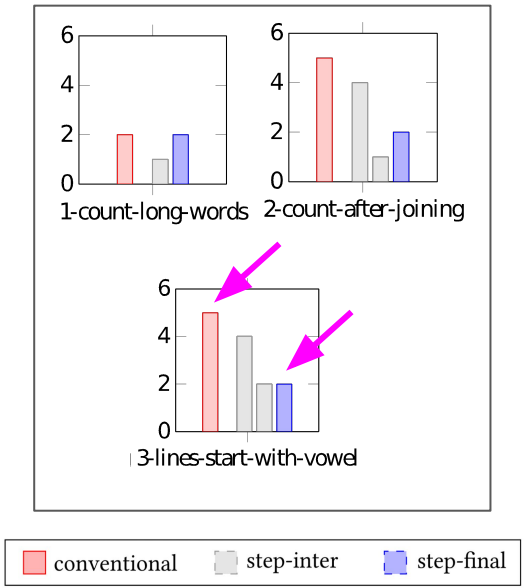
Task	Core task complexity		C/SSP
	C	SSP	
1-count-long-words	2	2	same complexity
2-count-after-joining	5	2	2.5x more complex
3-lines-start-with-vowel	5	2	2.5x more complex

Conventional

Simple-step pipeline

Result 2: incremental complexity reduction → lean core

"Paragraphs"
(mixed-content XML)



Task	Core task complexity		C/SSP
	C	SSP	
1-count-long-words	2	2	same complexity
2-count-after-joining	5	2	2.5x more complex
3-lines-start-with-vowel	5	2	2.5x more complex

Conventional

Simple-step pipeline

Conclusions

+

Future work

Conclusions

Refactoring a program as a small-step pipeline leads to a significant reduction in cyclomatic complexity (up to 2.5x).

(In XSLT/XPath programs.)

Reduction cyclomatic complexity → More readable, simpler programs.

> *Duaa Alawad, Manisha Panta, Minhaz F. Zibran, and Md Rakibul Islam. 2019. An Empirical Study of the Relationships between Code Readability and Software Complexity. CoRR abs/1909.01760 (2019). arXiv:1909.01760*

Future work

- More programming languages
- Bigger programs
- More complex XPath expressions (axes)
- Test with data “in the wild”

DocEng 2021 — Limerick, Ireland 2021-08-25

Small-step pipelines reduce the complexity of XSLT / XPath programs

https://gioele.io/p/doceng2021

Marcel Schaeben
m.schaeben@uni-koeln.de
Cologne Center for eHumanities
Köln, Germany

Gioele Barabucci
gioele.barabucci@ntnu.no
Norwegian University of Science and Technology
Trondheim, Norway