# Openly authored data pipelines

A demonstration for the WeCount Pluralistic Data Architecture

Antranig Basman

## Recapping goals of the pluralistic data architecture

- Allow communities to take ownership of data that relates to them and curate its relationships with data from other sources
- As related communities publish drafts of their own data, partners need to review the consistency of the different drafts and to ensure that appropriate versions of the data have been woven into their own view
- A set of tools capable of accepting data published in commonly understood formats (CSV, Excel, tabular data of other kinds) in commonly used venues (HTTP feeds, Google Sheets, GitHub, etc) and relating them together according to the community's chosen policies.
- Tracking the provenance of the data on a cell by cell basis, so that any community member can track the uses that have been made of their data, and also trace the ultimate source of any data that they make use of, regardless of how many hands it has passed through.

## A central problem in open authorship

What is the problem called?

“Materialised interception” - author A publishes a pipeline, author B wants to intercept in the middle of this pipeline without disturbing any of A’s names

- you then want the combined pipeline from A and B to be publishable in its own right and then further interceptable by C, etc.

## The Open Authorial Principle

“The design should allow the effect of any expression by one author to be replaced by an additional expression by a further author”

(Basman, Lewis, Clark, 2018 - https://github.com/amb26/papers/blob/master/onward-2016/onward-2016.pdf)

## Attempted solutions

Struggled with clumsy solutions to this for a while - mostly requiring some kind of “sequence alignment”, perhaps exploiting the sleazy new potential of JSON to preserve ordering

A big topic of Philip and my Salon 2020 paper - “Escaping the Prison of Style” - https://github.com/amb26/papers/blob/master/ccs-2020/ccs-2020.pdf

(paper includes a very verbose kind of pipeline in Python, using the Infusion “priorities” scheme - not a full solution since author B needs to rewrite some of A’s pipeline steps)

## Discovered solution

Emerged from 2016 thinking about “Plan to Abolish Invokers and Events” - [https://wiki.fluidproject.org/display/fluid/Plan+to+Abolish+Invokers+and+Events](https://wiki.fluidproject.org/display/fluid/Plan+to+Abolish+Invokers+and+Events)

- The basic framing is there, of a “new construct” unifying methods, listeners, and promise chains - but the central authorial move is missing, of how these things get composed together, and orchestrated in a modellised (materialised) idiom

- Started staring at the WeCount 3-element data pipeline again and, nudged by thinking from a notion of Jonathan Edwards, last month named an idea initially as “Vertical Chains”

## “Vertical Chains”

Idea is that there is a “vertical dimension” at each named point in the model, which hides an entire piece of submodel which from the outside appears to just have the name of its piece, but on the inside has the “gyre” structure

The “gyre” executes and produces its result, whether as a promise chain or something cheaper

The constructs *in* the gyre remain namable from the outside by using a special symbol - e.g. “joined^invent”

This idea is a sort of “enabling structure” for the solution but the actual solution mechanism is as follows:

## The solution - promotion during merging

The (options) merging process allocates a gyre on behalf of author A by promoting their artefact, if author B's interception looks like it will require it

This makes A's artefact from the point of view of B's pipeline structurally taller (but with the same information content) as when A wrote it

This process is reversible and transparent - an authoring tool can show to A's view of A, or B's view of A, together with their relationship

aka "Syncretic Promotion", "Enhouselling", etc.

## Author A’s pipeline - ODC/WeCount load and join

```
{
  type: "fluid.pipelines.WeCount-ODC",
  elements: {
    WeCount: {
      type: "fluid.fetchGitCSV",
      repoOwner: "inclusive-design",
      repoName: "covid-assessment-centres",
      filePath: "WeCount/assessment_centre_data_collection_2020_09_02.csv",
    },
    ODC: {
      type: "fluid.fetchGitCSV",
      repoOwner: "inclusive-design",
      repoName: "covid-assessment-centres",
      filePath: "ODC/assessment_centre_locations_2021_02_10.csv",
    },
    joined: {
      type: "fluid.forgivingJoin",
      left: "{WeCount}.data",
      right: "{ODC}.data",
      outerRight: true,
      outputColumns: {
        location_name: "ODC.location_name",
        city:          "ODC.city",
        website:       "ODC.website",
...
        "Accessible Parking":   "WeCount.Accessible parking",
        "Individual Service":   "WeCount.Personalized or individual service is offered",
        "Wait Accommodations":  "WeCount.Queue accommodations"
      }
    }
  }
}
```

https://github.com/inclusive-design/forgiving-data/blob/4c16bede4fe25073fee9e54c74e4a29c6a6b95fc/pipelines/WeCount-ODC.json5

## Author B’s pipeline - invent synthetic data

```
{
    type: "fluid.pipelines.WeCount-ODC-synthetic",
    parents: "fluid.pipelines.WeCount-ODC",
    elements: {
        joined: {
            type: "fluid.compoundElement",
            elements: {
                invent: {
                    type: "fluid.covidMap.inventAccessibilityData",
                    input: "{joined}.data",
                    seed: 0
                }
            },
            return: "invent"
        }
    }
}
```

https://github.com/inclusive-design/forgiving-data/blob/4c16bede4fe25073fee9e54c74e4a29c6a6b95fc/pipelines/WeCount-ODC-synthetic.json5

# Result after merging author A and author B's pipelines using structural promotion

B's overlaid artefact

A's promoted artefact

```
{
  "joined": {
    "type": "fluid.compoundElement",
    "elements": {
      "invent": {
        "type": "fluid.covidMap.inventAccessibilityData",
        "input": "{joined}.data",
        "seed": 0
      },
      "joined": {
        "type": "fluid.forgivingJoin",
        "left": "{WeCount}.data",
        "right": "{ODC}.data",
        "outerRight": true,
        "outputColumns": {
          "location_name": "ODC.location_name",
          "city": "ODC.city",
          "website": "ODC.website",
...
          "Accessible Parking": "WeCount.Accessible parking",
          "Individual Service": "WeCount.Personalized or individual service is offered",
          "Wait Accommodations": "WeCount.Queue accommodations"
        }
      }
    },
    "return": "invent"
  },
  "WeCount": {
    "type": "fluid.fetchGitCSV",
    "repoOwner": "inclusive-design",
    "repoName": "covid-assessment-centres",
    "filePath": "WeCount/assessment_centre_data_collection_2020_09_02.csv"
  },
  "ODC": {
    "type": "fluid.fetchGitCSV",
    "repoOwner": "inclusive-design",
    "repoName": "covid-assessment-centres",
    "filePath": "ODC/assessment_centre_locations_2021_02_10.csv"
  },
```

## Author C’s pipeline - output to file

Lives in A’s space of names, undisturbed by B

```
{
    type: "fluid.pipelines.WeCount-ODC-fileOutput",
    parents: "fluid.pipelines.WeCount-ODC",
    elements: {
        output: {
            type: "fluid.fileOutput",
            input: "{joined}.data",
            path: "outputData",
            value: "output.csv",
            provenance: "provenance.csv",
            provenanceMap: "provenanceMap.json"
        }
    }
}
```

https://github.com/inclusive-design/forgiving-data/blob/4c16bede4fe25073fee9e54c74e4a29c6a6b95fc/pipelines/WeCount-ODC-fileOutput.json5

## Merging together A and B’s pipelines

Requires A’s elementary “join” to be promoted to a “fluid.compoundElement” from the point of view of B

A’s name “joined” appears *inside* the compound element again as “joined”

Whilst in “Old Infusion” we do this by old-fashioned nesting, in “Future Infusion” this will occupy a “vertical dimension”, leaving ordinary nesting available for ordinary authors

## This is a mess in terms of previous constructs

This is clearly totally unacceptable from the point of view of functional programming - we have something with what looks like the “signature of a reducer” (mergePolicy) but which depends on all the surrounding structure

Also makes a mess of Infusion 2.x through Infusion 4.x

- can’t be a mergePolicy because there isn’t enough info in the signature, and we can’t configure it recursively

- can’t construct fresh components because it has already committed itself to be merging the options of a single component!

- just as well Infusion 5.x required a complete rewrite which is not really started, since otherwise we would just need to rewrite it again

## Mocked up in terms of Infusion 4.x

We can improve the underlying framework later, but the basic configuration elements for the pipeline are the way we want.

https://github.com/inclusive-design/forgiving-data/pull/2/files

# Infusion’s scoping blunder

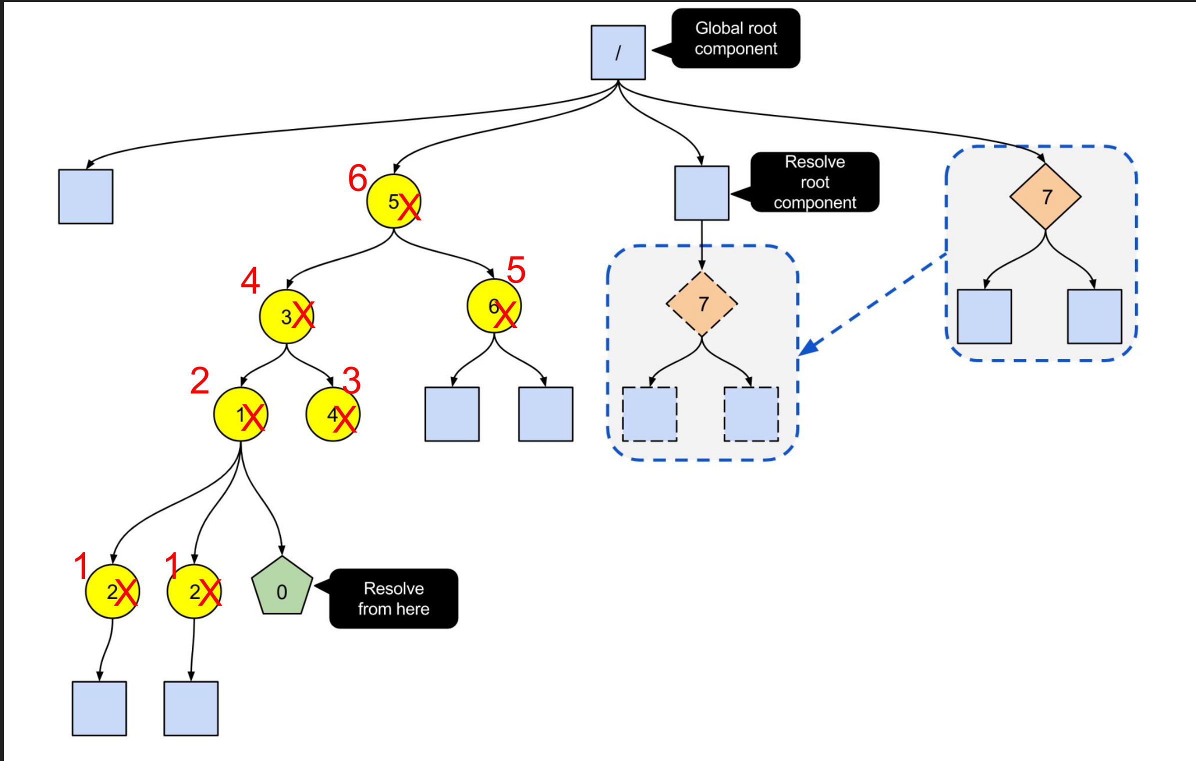
From https://docs.fluidproject.org/infusion/development/contexts

Has been in place since Infusion 1.3 in Dec 2010

Not clearly motivated by any particular usability concern

In the light of this new model is definitively incorrect - at least where we honour the “vertical dimension” by straightforward containment

Will be fixed in Infusion 6



## Related Issues

Justin’s very ancient mothballed pull request on the “Queued DataSource” (2014)

https://github.com/fluid-project/infusion/pull/566

This was abandoned because it was impossible to express the “interception” of the DataSource wrapping process within existing Infusion interception constructs

Was planned to accommodate this within “transforming promise chains” but these are awkward and flat, and as CCS2020 showed, still not sufficiently open

Marcel Weiher’s “Storage Combinators” - https://www.hpi.uni-potsdam.de/hirschfeld/publications/media/WeiherHirschfeld_2019_StorageCombinators_AcmDL_Preprint.pdf - referenced in his own 2020 Salon paper - were considered out of scope for the same reason

“Queued DataSource” was actually a storage combinator in Weiher’s sense - these kinds of solutions can now be revisited