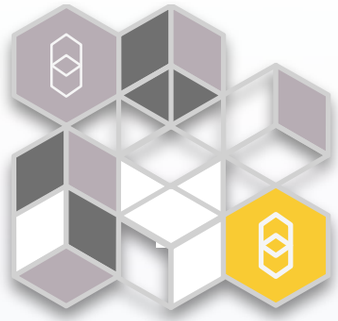
omii-uk
www.omii.ac.uk

myGrid



Advanced Taverna
.. and introducing Taverna 2

Stian Soiland

NBIC, Wageningen University
2007-12-12

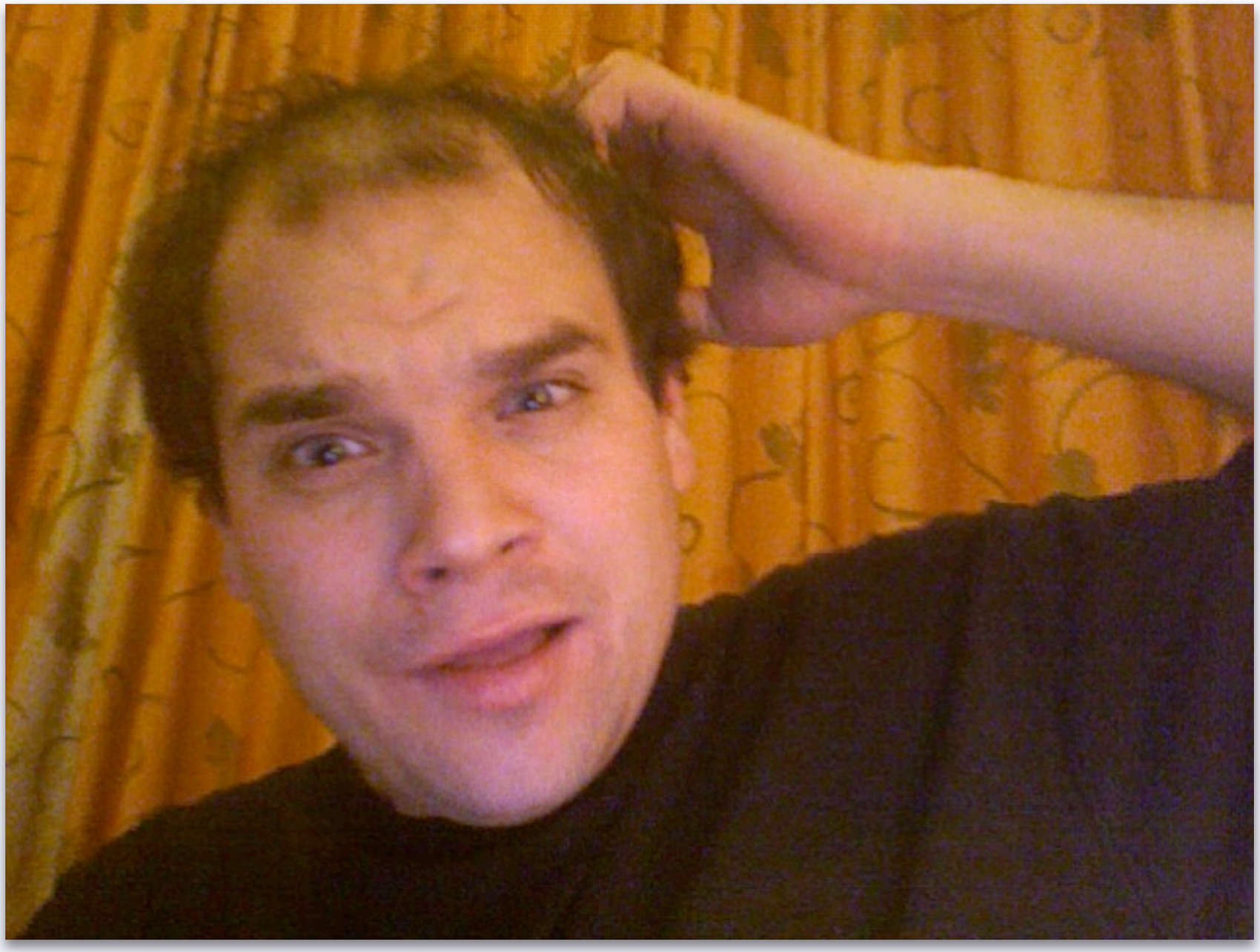
Powerpoint warning

- This presentation is not using PowerPoint
  - OK, but it's using Keynote from Apple
  - But that's not the point
- I will attempt my best to talk about our subject freely
  - The slides will be used mostly for illustrations
  - I might switch the screen to live Taverna
- I will use the **white board**
- I will use **my voice** (and hopefully my brain)
- .. but I might not be consistent in following this mantra

Attention warning

- If you just downloaded these slides from the Internet
- .. or you only look at the projector screen
- Then:
  - You won't hear what I'm saying
  - You won't see what I'm drawing
  - --> You won't understand much
- Now where's that clipart of the guy scratching his head..?

The head scratching guy



Overview #1

Iteration

- Implicit iteration
- Errors
- Error recovery
  - Retry
  - Failover (Alternate)
  - Loops?
  - Iteration strategies

Overview #2

Taverna 1 limitations

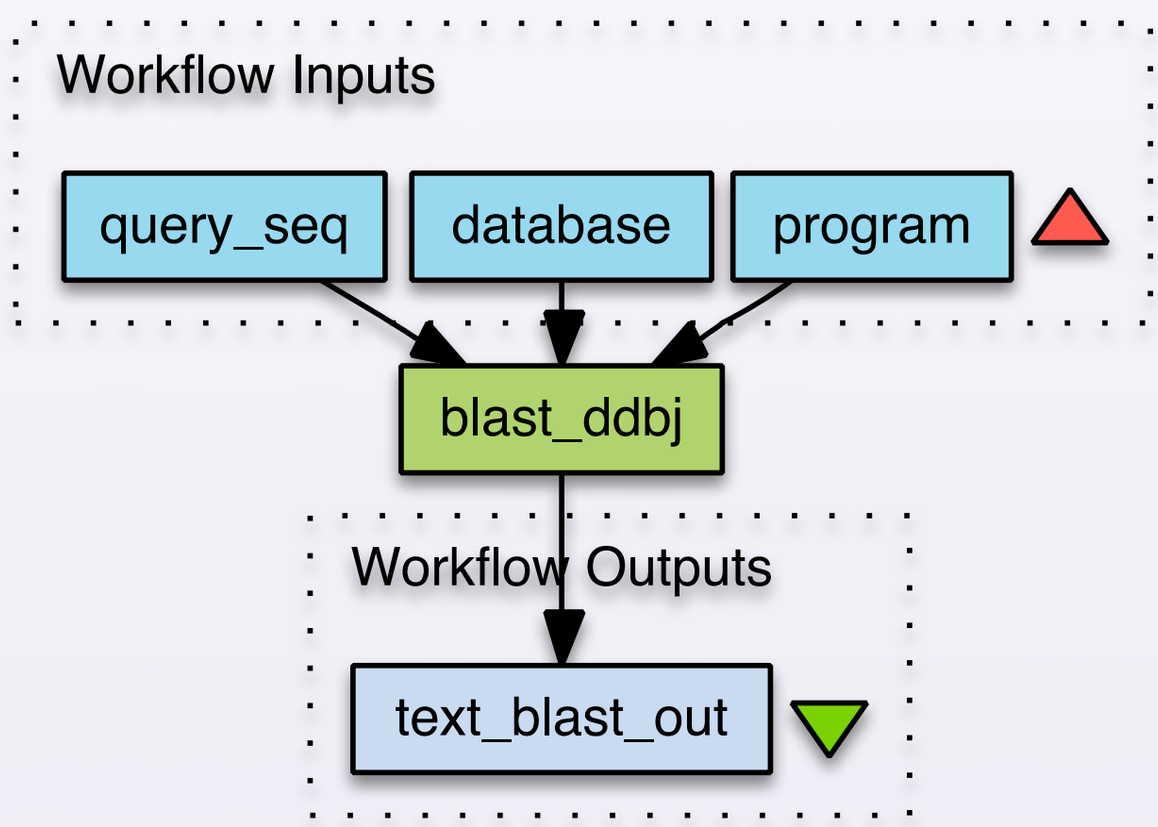
- Long pipeline
- Large data

Taverna 2 improvements

- Streaming
- Data types
- Data references

Iteration: Simple workflow

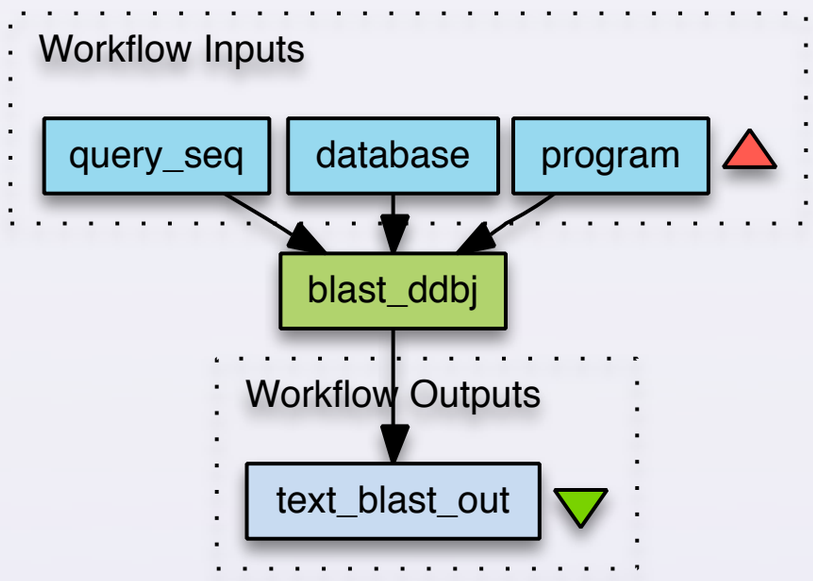
Assume a very simple workflow:



Paul Fisher, http://www.myexperiment.org/workflows/23

Input ports' syntactic type

All inputs of blast_ddbj have *syntactic type* 'text/plain'

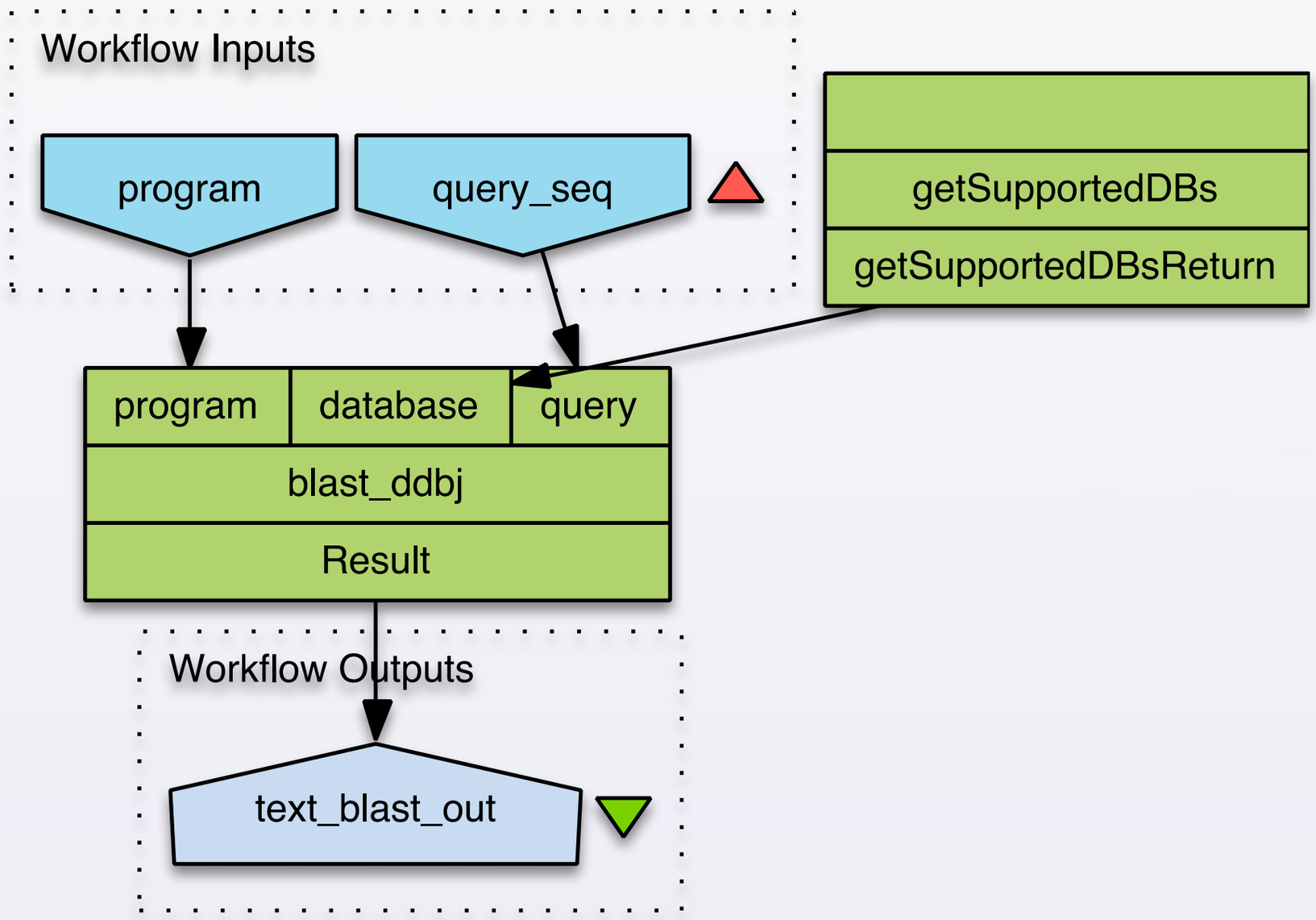


List type

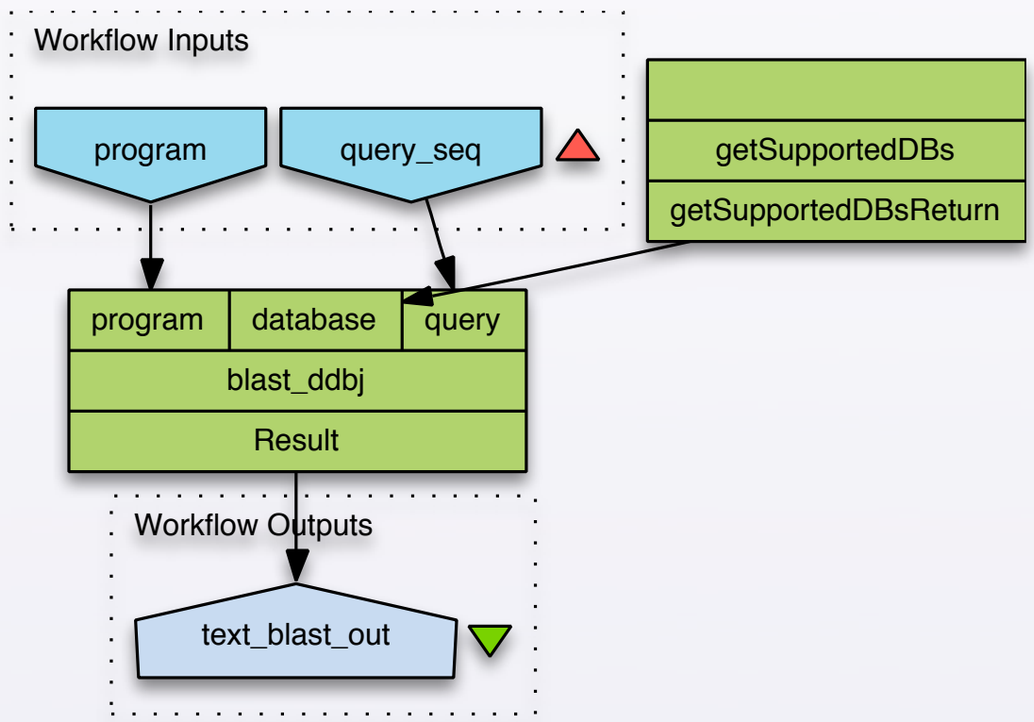
- But here's a service that outputs l('text/plain')

- The l() means a return of **list of strings**

Implicit iteration



Implicit iteration results



- Workflow output is now a **list** of blast reports, one for each database
- See whiteboard for Stian's scribblings

in t2: list depth

- **l**('text/plain') --> depth **1**
- **l(l**('text/plain')) --> depth **2**
- ..

and of course..

- text/plain --> depth **0**

in t2: Error documents

- What if the third element of an iteration fails?

t2: Propagation of errors

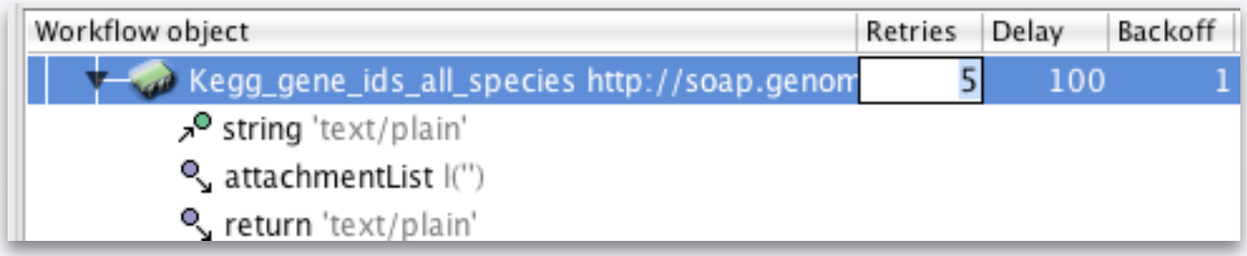
- Error documents float through the workflow as data

t1: Error recovery

- Taverna has two main ways to recover from errors

Retries

- Can specify how many times to retry (Retries)
- And how much to sleep before trying again (in milliseconds) (delay)



- Advanced: Can incrementally increase the delay (Backoff)
  - Stay close to 1, say 1.1, to avoid exponential growth

Alternate processors

- Other services can be dragged onto a processor as an alternate

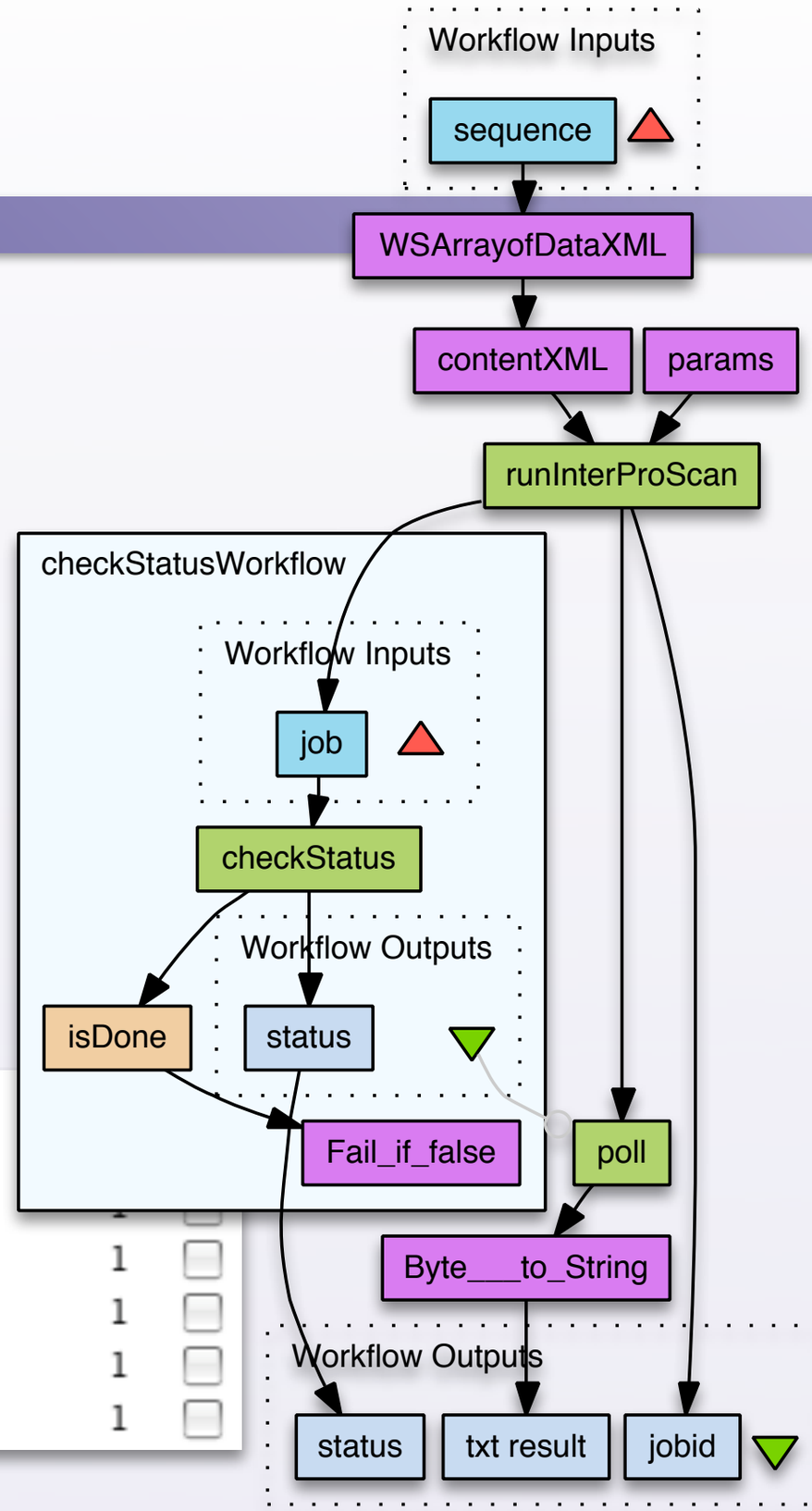
- Special case: Abstract processor
  - Non-invocable processor
  - Can add implementations as alternates later

t2: Activities and processors

- Taverna 2 separates between:
  - Processor (part of workflow)
  - Activity (invokes service)
- The processor performs the iteration, retries, failovers, etc
- The service contacts a concrete service (say at a SOAP endpoint)
- Processor can have 0 or more activities
- Abstract processor will mean no activities

Loop hack

- The retry mechanism can be used as a looping mechanism:
- nested workflow fails except when status isDone is true

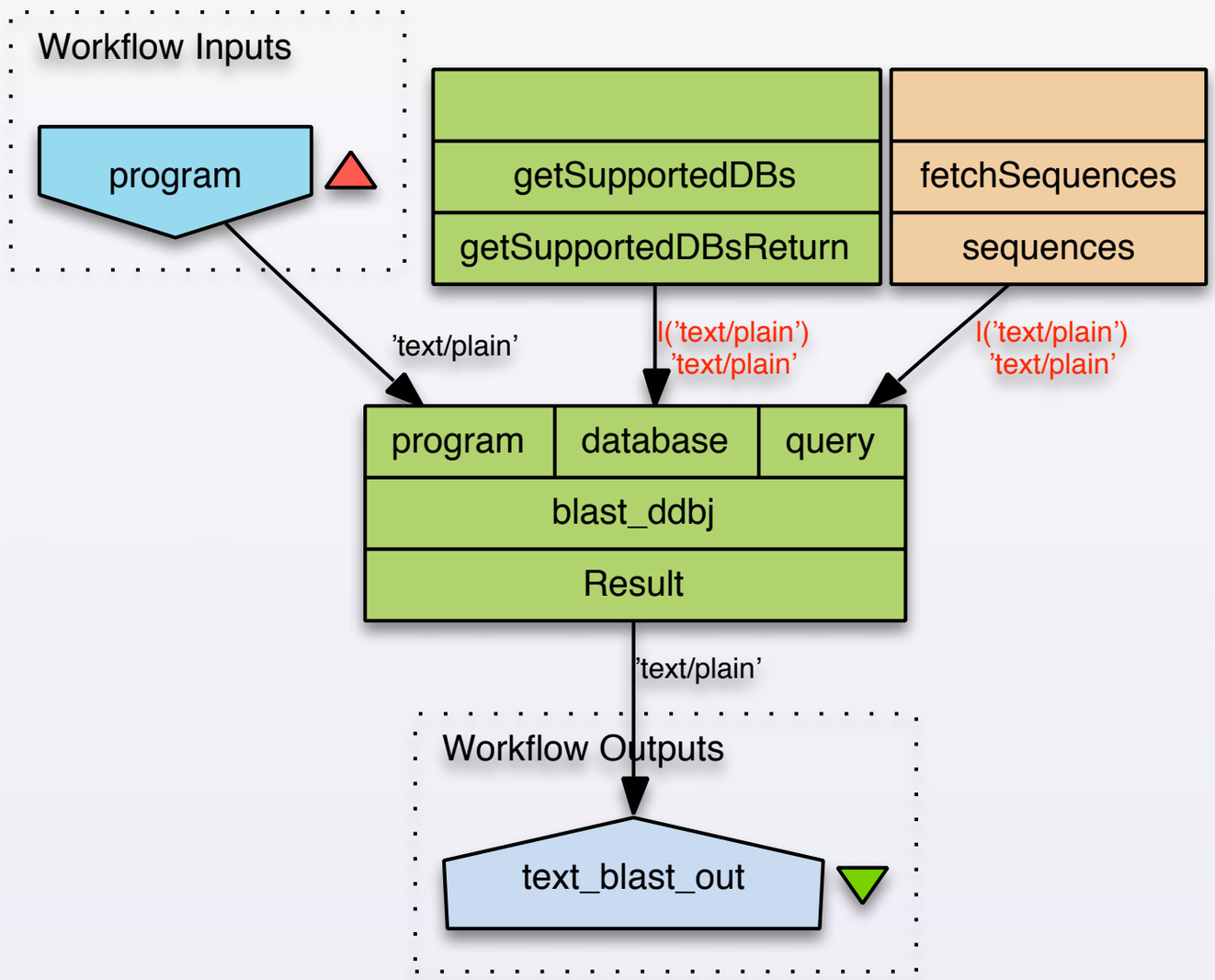


Byte___to_String	0	0	1	
contentXML	0	0	1	
WSArrayofDataXML	0	0	1	
params	0	0	1	1
runInterProScan http://www.ebi.ac.uk/Tools/w	0	0	1	1
poll http://www.ebi.ac.uk/Tools/webservices/	0	0	1	1
checkStatusWorkflow	10	30000	1	1

Duncan Hull, http://www.myexperiment.org/workflows/4

Iteration strategies

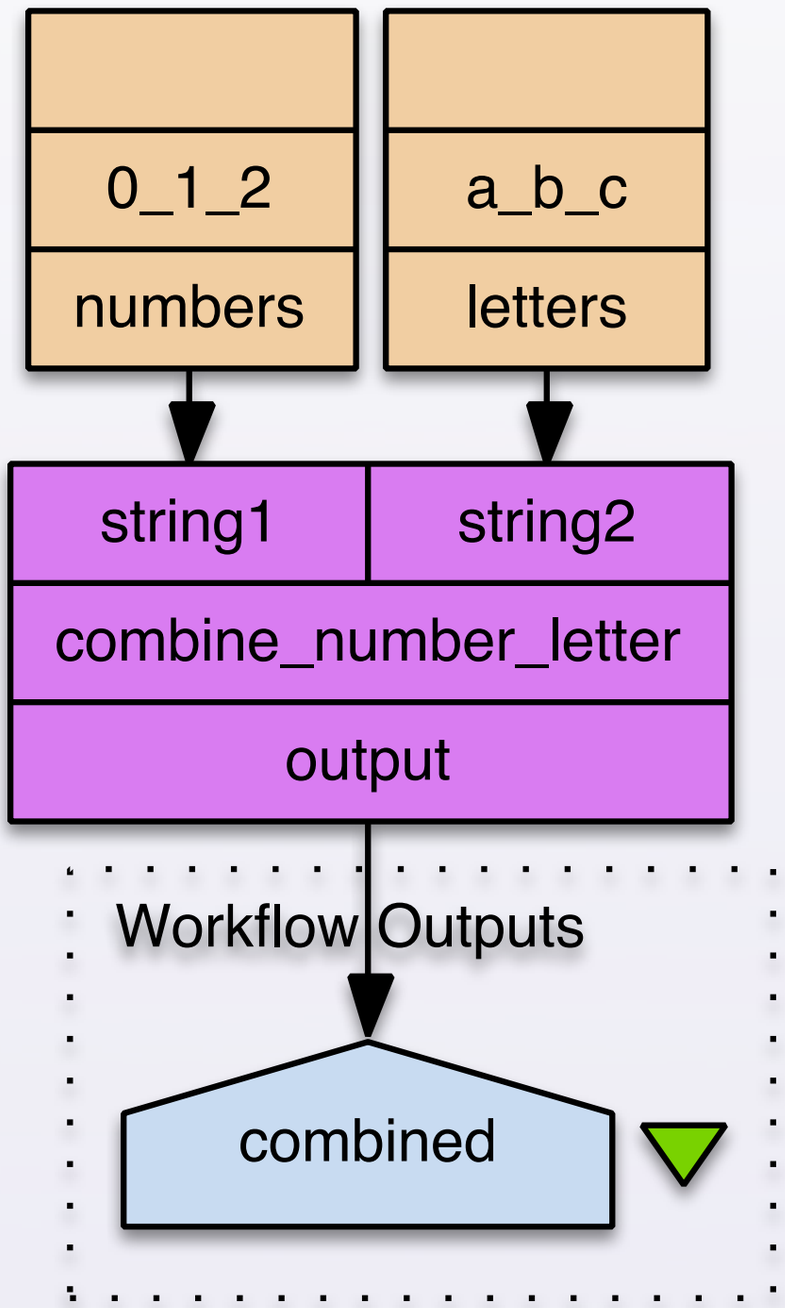
- More complex iteration:



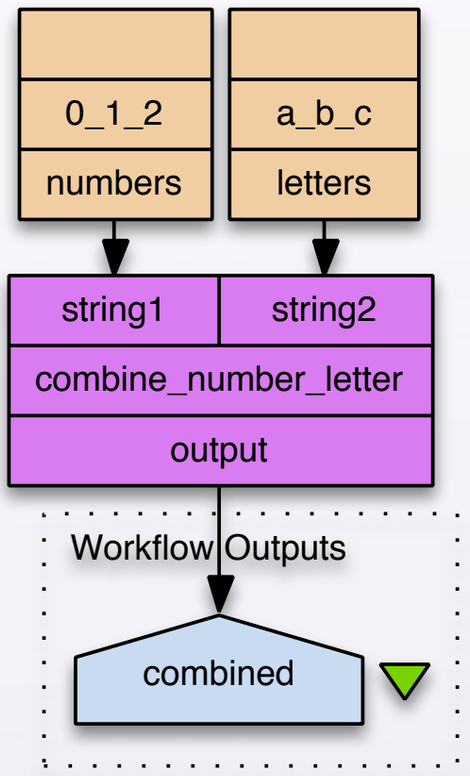
- But which one is iterated first?

Try with a simpler workflow

- Dummy workflow that just generates strings:



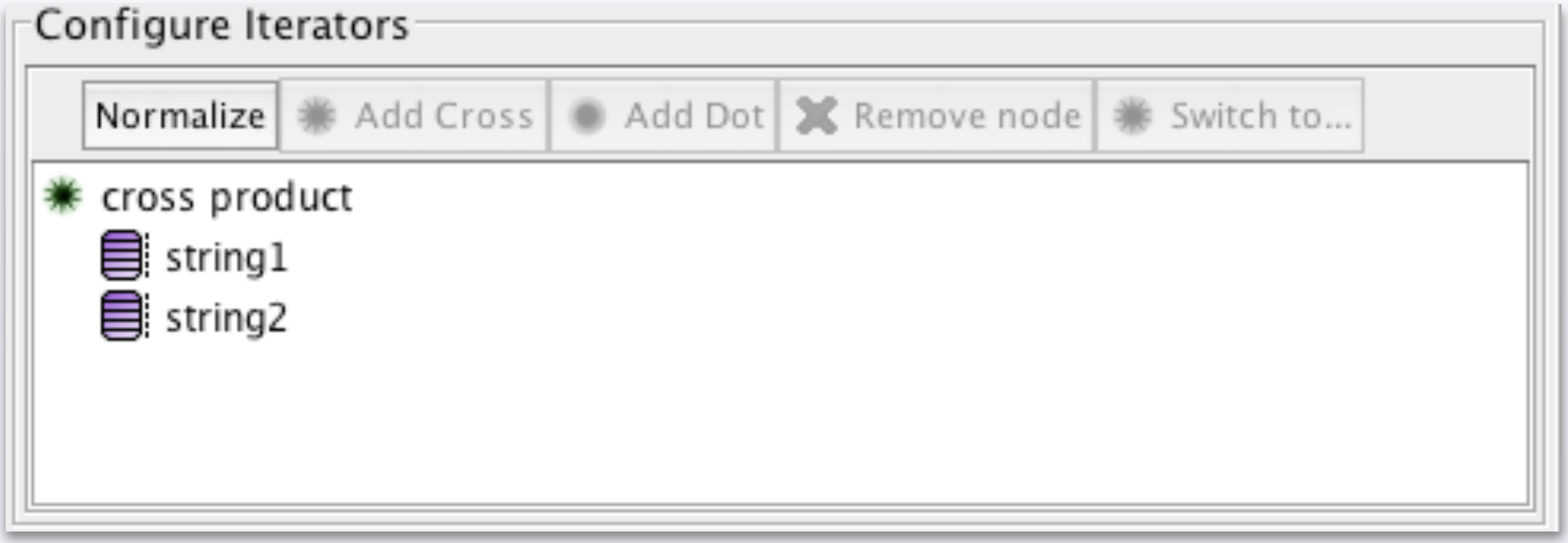
Default iteration strategy



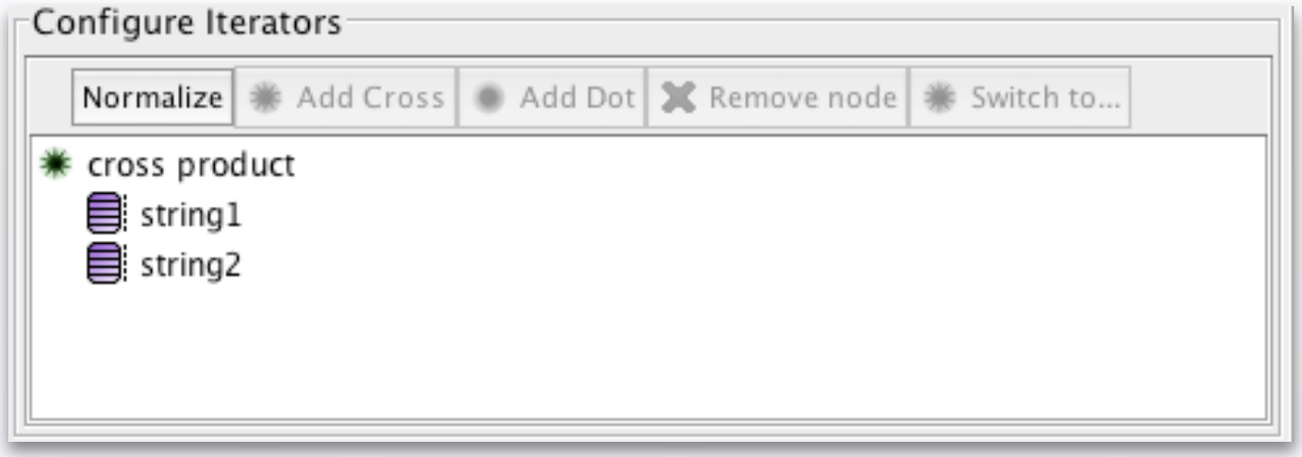

```
[ [a0, a1, a2],
  [b0, b1, b2],
  [c0, c1, c2]
]
```

But I want [a0, b1, c2]!

- Go to **Metadata** and click **Create iteration strategy:**



Control iteration order




```
[ [a0, b0, c0],
  [a1, b1, c1],
  [a2, b2, c2]
]
```

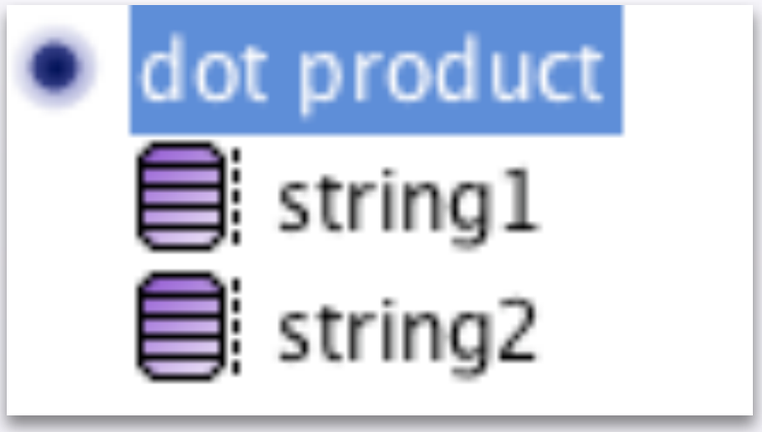
- .. and get this time:

- compared with the old undetermined order:

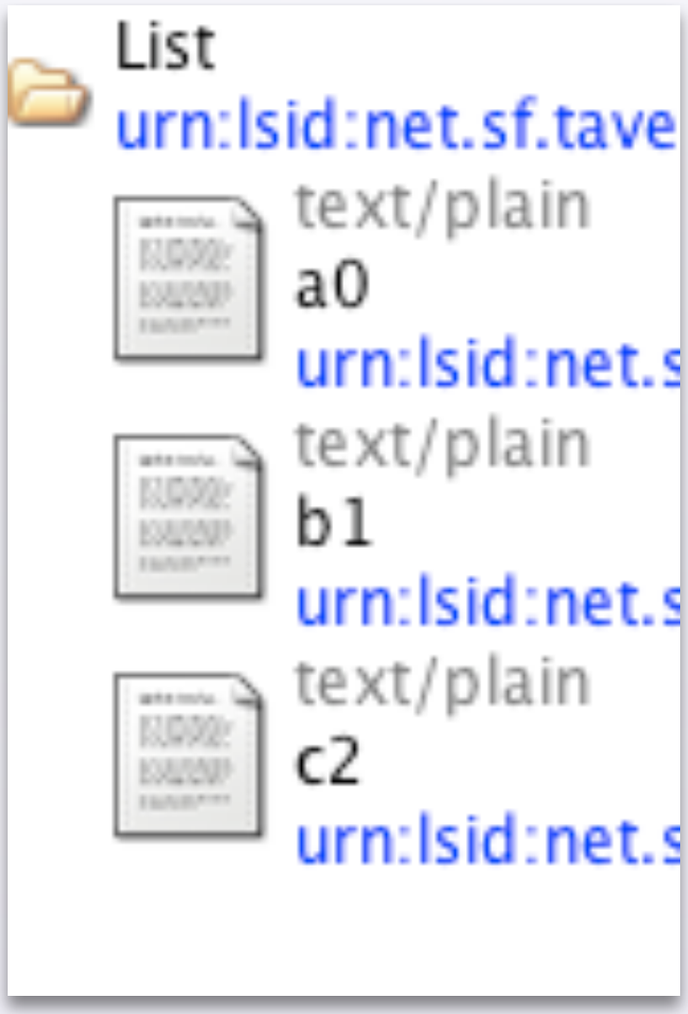
```
[ [a0, a1, a2],
  [b0, b1, b2],
  [c0, c1, c2]
]
```

But I still want [a0, b1, c2]!

- OK, ok, click on the **Switch to..** button and make a **dot product**:

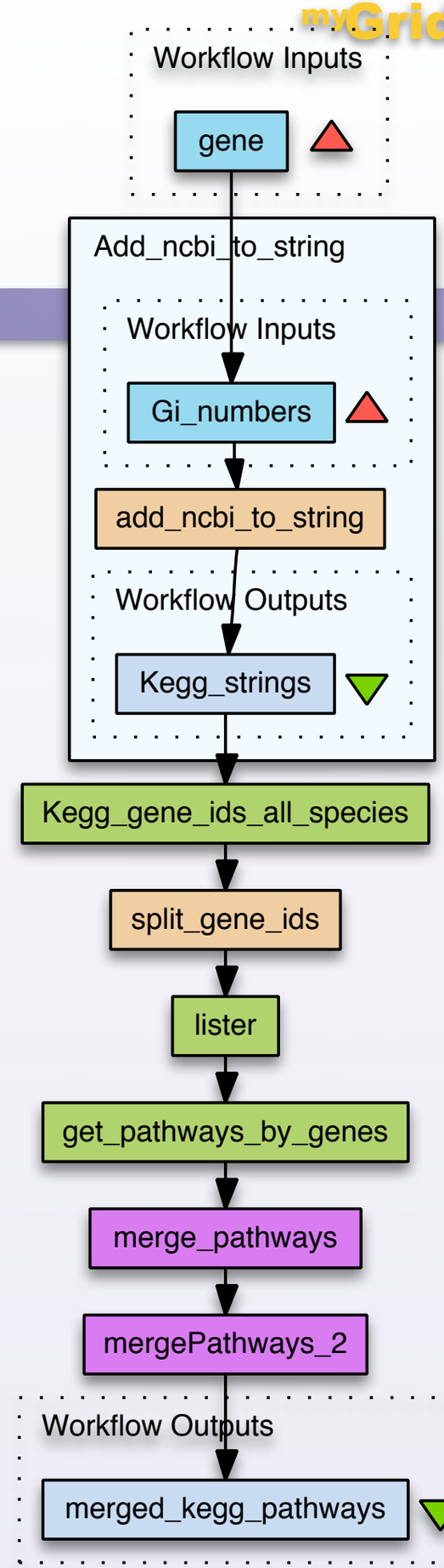


- Which gives us..



Workflow as a pipeline

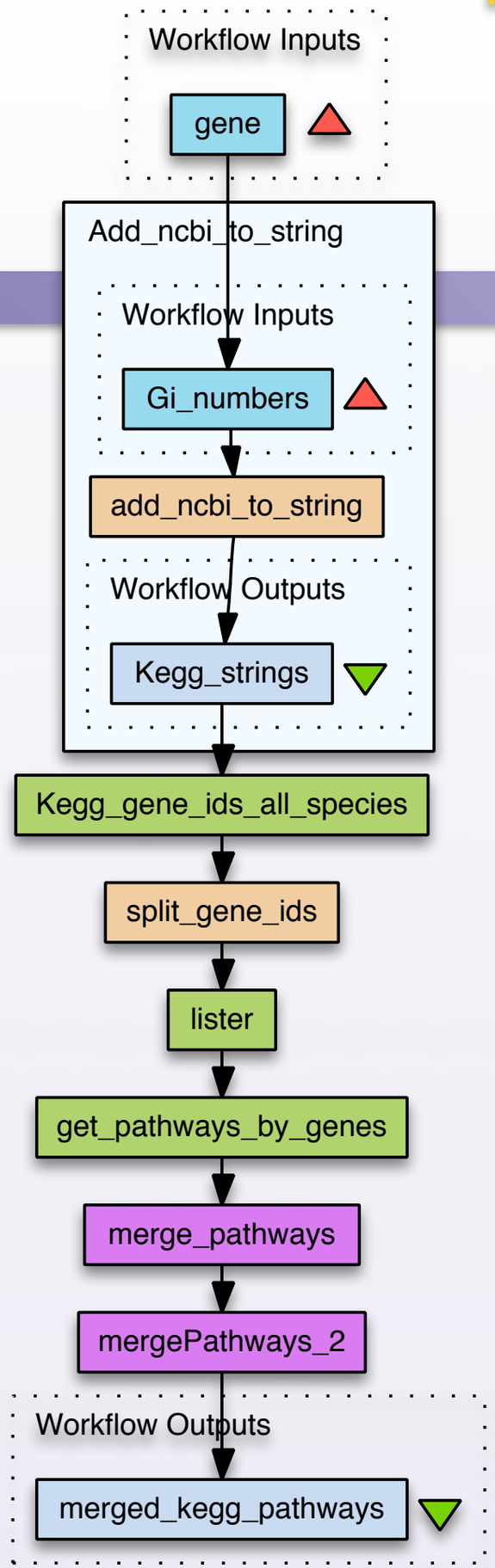
- OK, this one I did not make up:



Paul Fisher, http://www.myexperiment.org/workflows/15

Slow iteration

- Watch the **whiteboard** as Stian explains why this could take a while with 10-15 genes as inputs



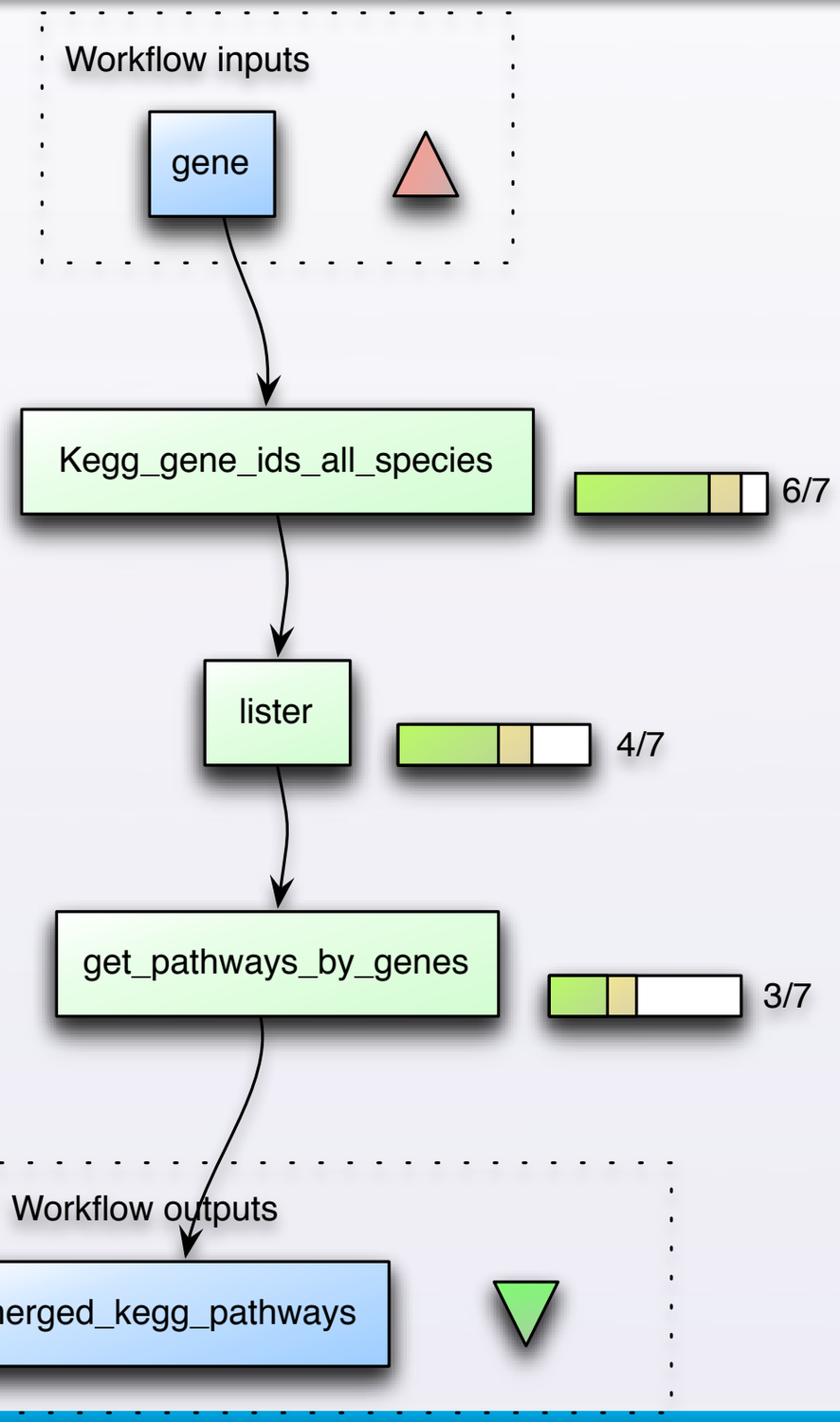
Quick hack: More threads

- Increase threads slightly for the slow ones

Workflow object	Retries	Delay	Backoff	Threads	Critical
Workflow inputs					
gene					
Workflow outputs					
merged_kegg_pathways					
Processors					
split_gene_ids	0	0	1	1	
merge_pathways	0	0	1	1	
mergePathways_2	0	0	1	1	
get_pathways_by_genes http://soap.genome.j	0	0	1	3	
Kegg_gene_ids_all_species http://soap.genom	0	0	1	3	
Add_ncbi_to_string	0	0	1	1	
lister http://phoebus.cs.man.ac.uk:8081/axis/	0	0	1	3	

t2: Streaming

- See **whiteboard** for manual iteration by Stian



t2: Services can stream too

- Services that returns lists, such as BioMart
  - Create single data items for each line
  - See whiteboard
- Can come out of order
- Infinite lists
  - Instruments
  - Updates
  - “Push”

Also slow: Big stuff

- t1: Anything bigger than a few MBs can be very slow
  - Mainly because it's stored in memory
  - Sent up and down several times over slow network connection

Big stuff over the wire

- t1 workarounds:
  - Instead of:
    - GATTGAGAGACCCAGAGG….
  - return a service-local identifier:
    - sequence:23231
  - See **whiteboard**
  - Or..re-use existing identification schemes
    - URI (Might even be fetchable)
    - LSID (Could be resolvable)
    - Specialised scheme (Uniprot, pubmed, etc)

Quick hack: Data proxy

- myGrid's Data proxy wraps WSDL services
- Extracts selected elements of the schema and stores them to disk
- See **whiteboard**

t2: Big stuff

- t2: Several data managers can store data in:
  - memory
  - files
  - on another machine
- t2: Lists and errors are also data

t2 data types (Entities)

- Literal
  - Integer, Float, Double, etc.
  - Small strings (< 80 chars)
- Data document
  - With references to stored bytes/strings
- Error document
  - Error message
- List
  - Containing any of the above

t2: Reference schemes

- Built-in support for references
- Multiple types of references (even for same data)
- A reference might be local for:
  - Program
  - User
  - Machine
  - Network
  - Service

There, did you see?

- The powerpoint mentality returned
- That list is not interesting at all, why spend 50% of screen on that?
- I'll try again:

t2: Reference schemes

- Data documents are just pointers to the real data
- References of any type
- Services might produce and consume references
- .. or still do old-style data on the wire

t2: Entity identifiers

- We try to practice what we preach, so t2 also uses references internally
- Lists, data documents and error documents have identifiers
- So a list of data documents don't actually contain the data documents, just the identifiers of those documents

t2: List with identifiers

The gory details.. can you spot the references?

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ent:entityList xmlns:ent="http://taverna.sf.net/t2/cloudone/bean/">
   <identifier>
      urn:t2data:list://021a6d32-5aba-4f34-bd32-c6989b724496/
       69350508-83c3-4e7c-bd11-e97e7f22507c/1
   </identifier>
   <entity>
      urn:t2data:ddoc://021a6d32-5aba-4f34-bd32-c6989b724496/
            a0fed96c-181f-40b9-9f22-13225b3b7c34
   </entity>
   <entity>
      urn:t2data:ddoc://021a6d32-5aba-4f34-bd32-c6989b724496/
            4ebf49cd-74d8-4da5-9173-d0ca4a3c5c5d
   </entity>
</ent:entityList>
```

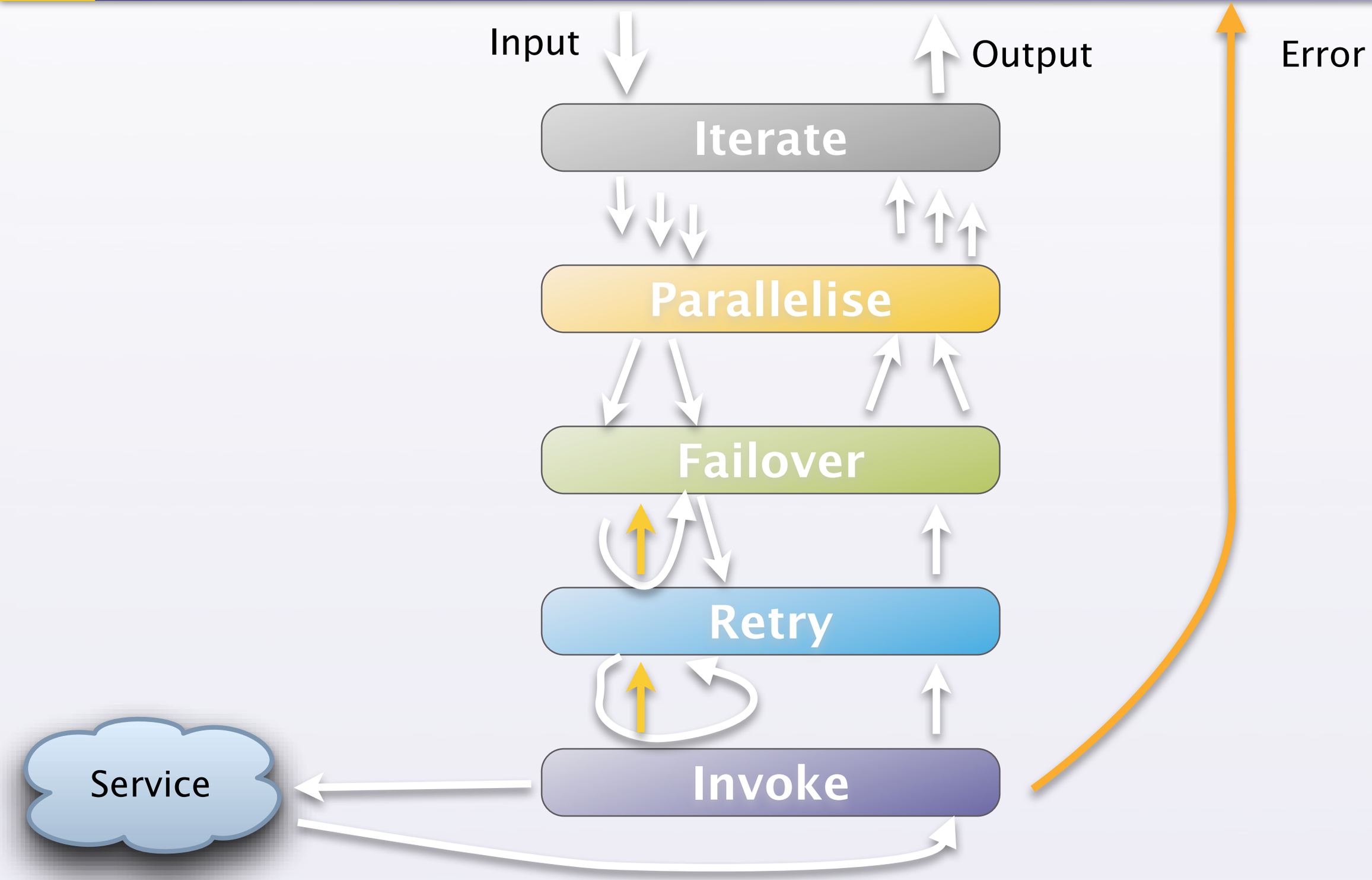
t2 future: Multiple enactors

- A workflow can run on a combination of machines
- Enactors communicate using a p2p protocol
- Entities and data are shared if needed
- Cooperate to get access to resources

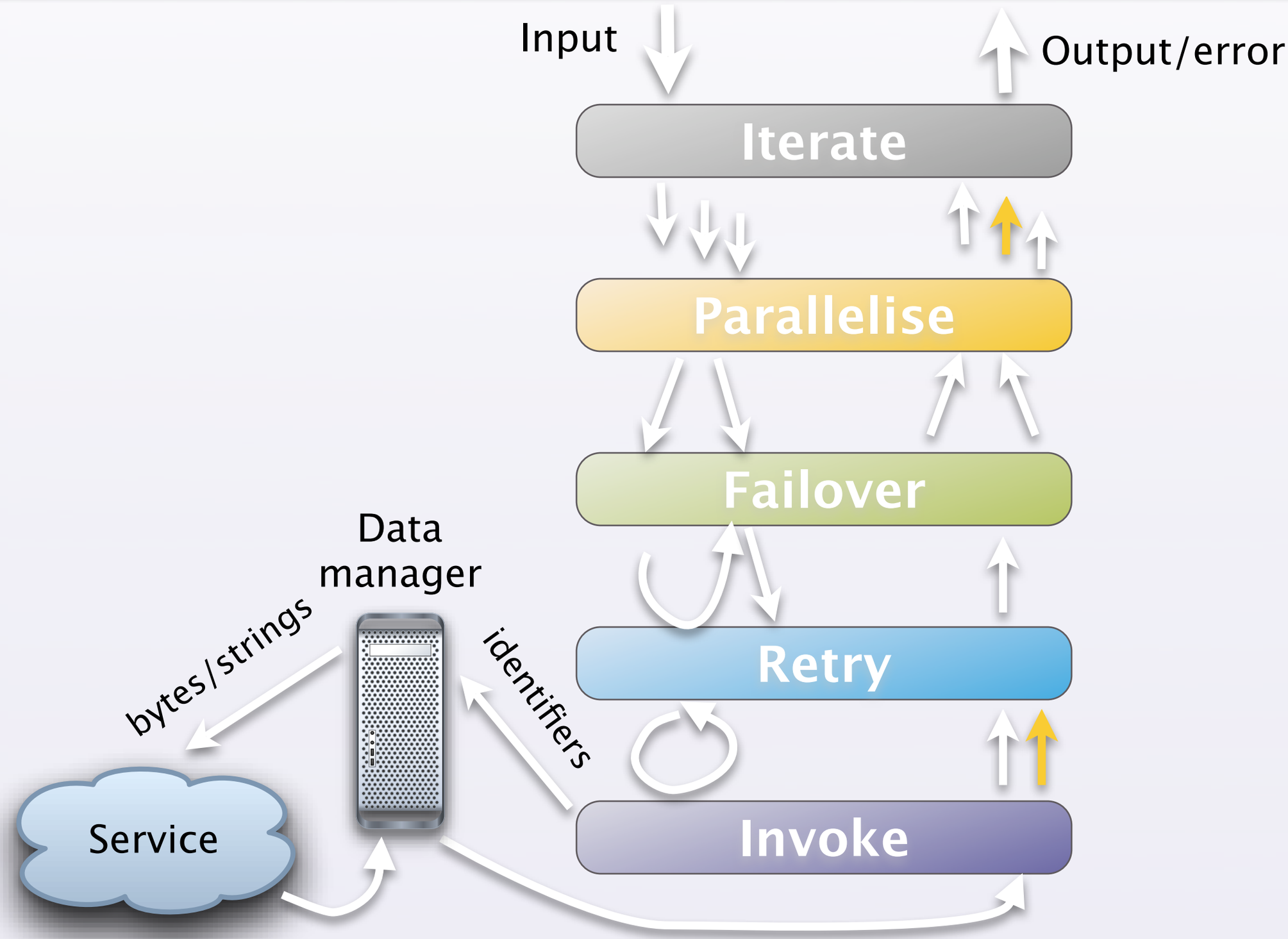
Recap of workflow

- Processor contains activities
- Processor does:
  - Implicit iteration
  - Delay and retry
  - Failover to alternate activity
  - Map to selected activity
- Activity does:
  - Invoke the service
  - Register result data

Taverna 1 invocation stack



Taverna 2 default stack



t2: stack can be changed

- For advanced users..
  - Failover first, then retry
  - Failover for the full iteration
  - Look up services at run-time
  - Parallelise on several computers
  - … anything

Summary

Where do we go next with t2?

- Taverna 1.7 will be relased 2007-12-17
  - Includes a t2 “taster” plugin
  - New available services palette
- More updates for the t2 plugin in 2008
- Improving/hardening APIs to ease the transition for projects like BioMoby
- Update GUI for new features
- Grid and security
- Multiple enactors (p2p)