Collection-Oriented Scientific Workflows for Integrating and Analyzing Biological Data

Shawn Bowers[1]
Timothy McPhillips[1]
Bertram Ludaescher[1,2]

[1] UC Davis Genome Center
[2] Dept. of Computer Science
University of California, Davis

Cutting to the chase: What are Scientific Workflows?

- **Probably the single-most important concept you hear about @DILS'06**
- **Attempto-Plain-English Definition:**
  - ***SWFs := System designs and/or executable programs/scripts***
  - ***… aiming to solve complex scientific data integration, analysis, management, visualization tasks***
    - in plainer English: doing hard and/or messy stuff …
  - ***… while doing it in a scientist-friendly way***
    - that is: making it **look** easy
  - ***… with the ultimate goal to***
  - ***… do new, more, and better (e-)Science,***
  - ***… faster!***
- **In short: SWFs are nothing less than MIRACLE-IT® to make scientists (biologists, physicists, ...) happy.**

  Bio to CS/IT guy: "Please MIRACLE-IT® !" and "MIRACLE-IT®-Happen-**NOW**!!"
- **Attempto-DL Definition:**

$$\mathsf{HappyScientist} \equiv \mathsf{Scientist} \sqcap \forall \mathsf{hasWF}.\mathsf{HappyWF}$$
$$\mathsf{HappyWF} \equiv \mathsf{Workflow} \sqcap \mathsf{Reusable} \sqcap \mathsf{Configurable} \sqcap \mathsf{Efficient} \sqcap \ldots$$

Ex: A Happy Fusion Simulation Workflow

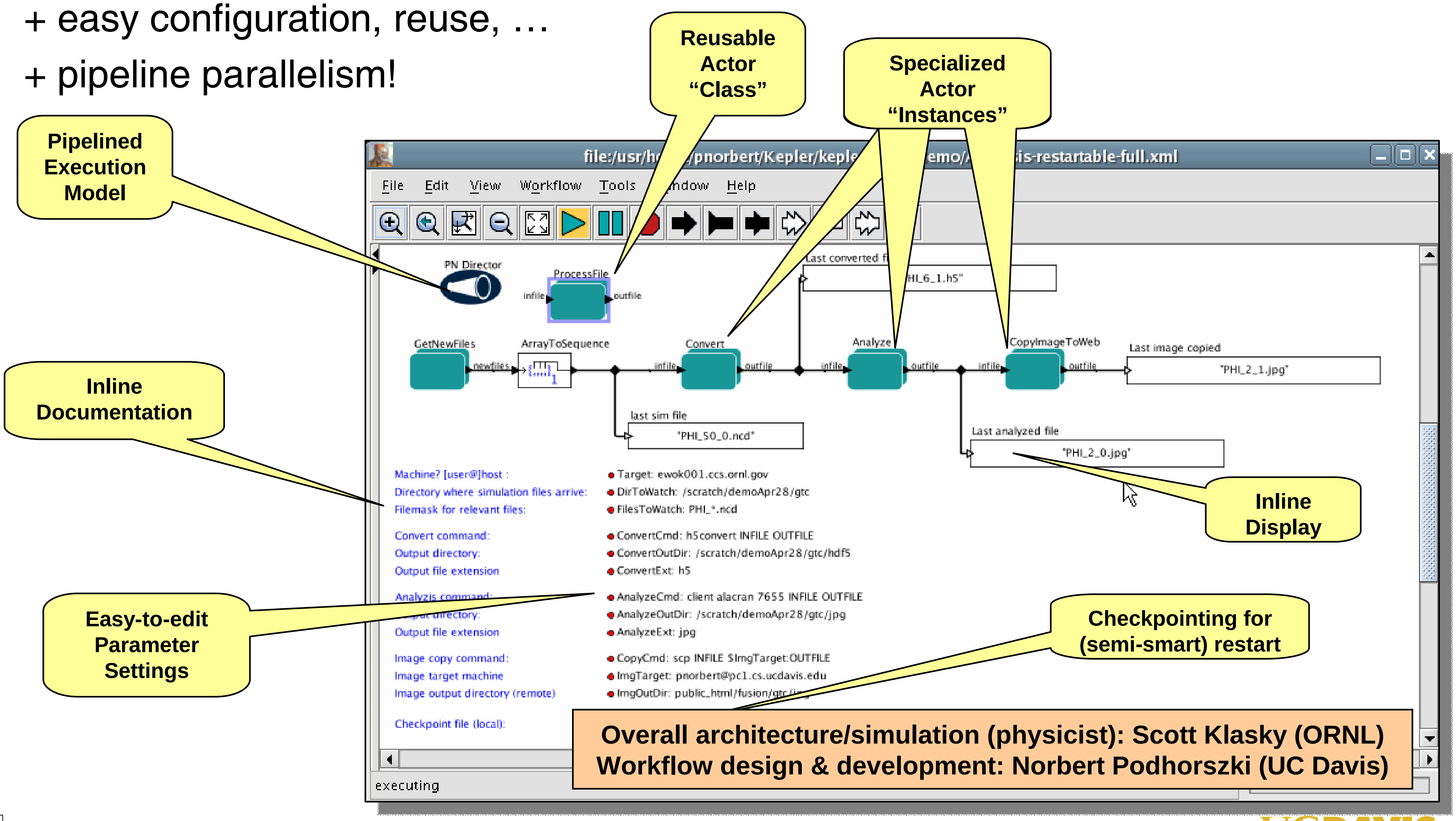
Subspecies/Variety: *“Plumbing WF” (flux-laboris plumbiensis)*

- **Implements concurrent analysis pipeline (@2ndary cluster):**

  Tasks: *convert*; *analyze*; *copy-to-Web-portal* (makes scientists **really** happy!)

  + easy configuration, reuse, …

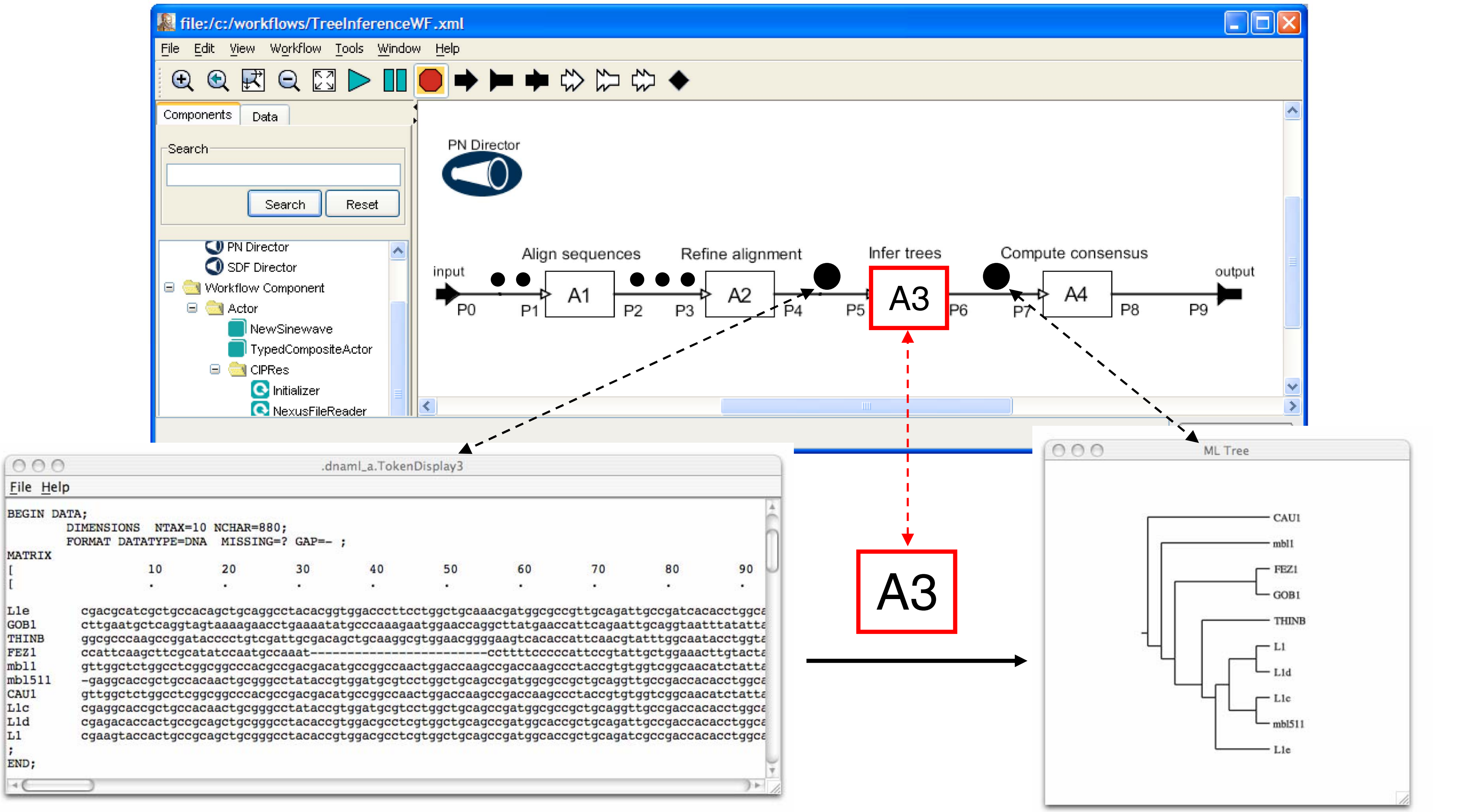
  + pipeline parallelism!



Machine? [user@]host :	Target: ewok001.ccs.ornl.gov
Directory where simulation files arrive:	DirToWatch: /scratch/demoApr28/gtc
Filemask for relevant files:	FilesToWatch: PHI_*.ncd
Convert command:	ConvertCmd: h5convert INFILE OUTFILE
Output directory:	ConvertOutDir: /scratch/demoApr28/gtc/hdf5
Output file extension	ConvertExt: h5
Analyzis command:	AnalyzeCmd: client alacran 7655 INFILE OUTFILE
…directory:	AnalyzeOutDir: /scratch/demoApr28/gtc/jpg
Output file extension	AnalyzeExt: jpg
Image copy command:	CopyCmd: scp INFILE $ImgTarget:OUTFILE
Image target machine	ImgTarget: pnorbert@pc1.cs.ucdavis.edu
Image output directory (remote)	ImgOutDir: public_html/fusion/gtc/…
Checkpoint file (local):	

Overall architecture/simulation (physicist): Scott Klasky (ORNL)
Workflow design & development: Norbert Podhorszki (UC Davis)

Ex: Pipelined workflow for inferring phylogenetic trees



```
BEGIN DATA;
        DIMENSIONS  NTAX=10 NCHAR=880;
        FORMAT DATATYPE=DNA  MISSING=? GAP=- ;
MATRIX
[                  10        20        30        40        50        60        70        80        90
[                  .         .         .         .         .         .         .         .         .

Lle       cgacgcatcgctgccacagctgcaggcctacacggtggacccttcctggctgcaaacgatggcgccgttgcagattgccgatcacacctggca
GOB1      cttgaatgctcaggtagtaaaagaacctgaaaatatgcccaaagaatggaaccaggcttatgaaccattcagaattgcaggtaatttatatta
THINB     ggcgcccaagccggatacccctgtcgattgcgacagctgcaaggcgtggaacggggaagtcacaccattcaacgtatttggcaatacctggta
FEZ1      ccattcaagcttcgcatatccaatgccaaat------------------------cctttccccattccgtattgctggaaacttgtacta
mbl1      gttggctctggcctcggcggcccacgccgacgacatgccggccaactggaccaagccgaccaagccctaccgtgtggtcggcaacatctatta
mbl511    -gaggcaccgctgccacaactgcgggcctataccgtggatgcgtcctggctgcagccgatggcgccgctgcaggttgccgaccacacctggca
CAU1      gttggctctggcctcggcggcccacgccgacgacatgccggccaactggaccaagccgaccaagccctaccgtgtggtcggcaacatctatta
Llc       cgaggcaccgctgccacaactgcgggcctataccgtggatgcgtcctggctgcagccgatggcgccgctgcaggttgccgaccacacctggca
Lld       cgagacaccactgccgcagctgcgggcctacaccgtggacgcctcgtggctgcagccgatggcaccgctgcagattgccgaccacacctggca
L1        cgaagtaccactgccgcagctgcgggcctacaccgtggacgcctcgtggctgcagccgatggcaccgctgcagatcgccgaccacacctggca
;
END;
```

Aligned sequences "token"

Phylogenetic tree "token"

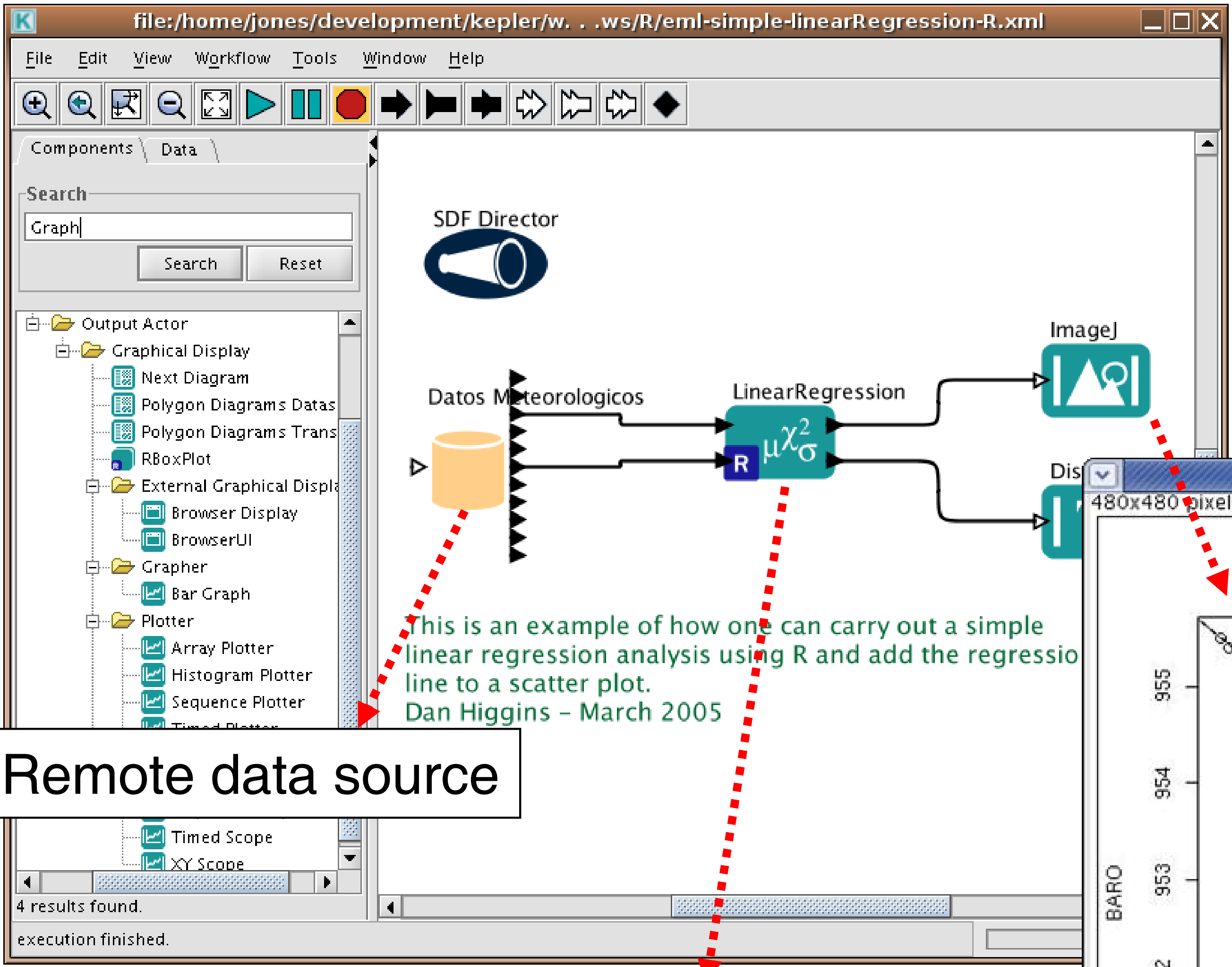
UCDAVIS UNIVERSITY OF CALIFORNIA

What about scripts instead of SWFs?

- **Why not just use …**
  - MIRACLE-Perl®
  - or MIRACLE-Python®
  - or MIRACLE-BPEL4WS® ???
- **Perl/Python in the hand of a gifted** (and in the case of Perl: masochistic) **programmer are hard to beat …**

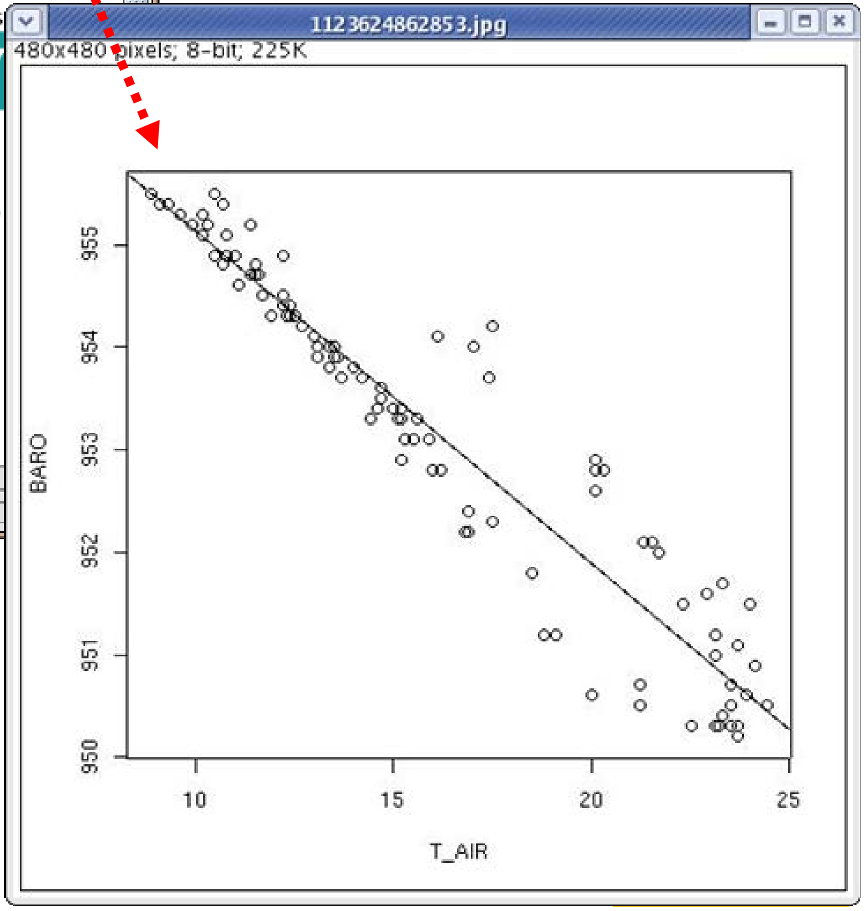
- **… but (MIRACLE-) Scientific Workflows offer some new features:**
  - parameter configuration, parameter studies
  - actor (component)-oriented workflow design [Bowers-Ludaescher-ER'05]
    - component and workflow reuse & repurposing
  - semantic extensions (smart search/link/…) [Bowers-Ludaescher-QLQP'06]
  - data (and workflow) provenance support ([Altintas-*et-al*], [McPhillips-*et-al*]) @ IPAW'06
    - explain data dependencies/lineage, debug "strange" results, smart rerun,…
  - data-, task-, **pipeline**-parallelism
  - comprehensibility, documentation
- **e.g. check out demos by Kepler, Taverna, …**

Kepler SWF using remote datasets, 3rd-party software …



This is an example of how one can carry out a simple linear regression analysis using R and add the regressio line to a scatter plot.
Dan Higgins - March 2005

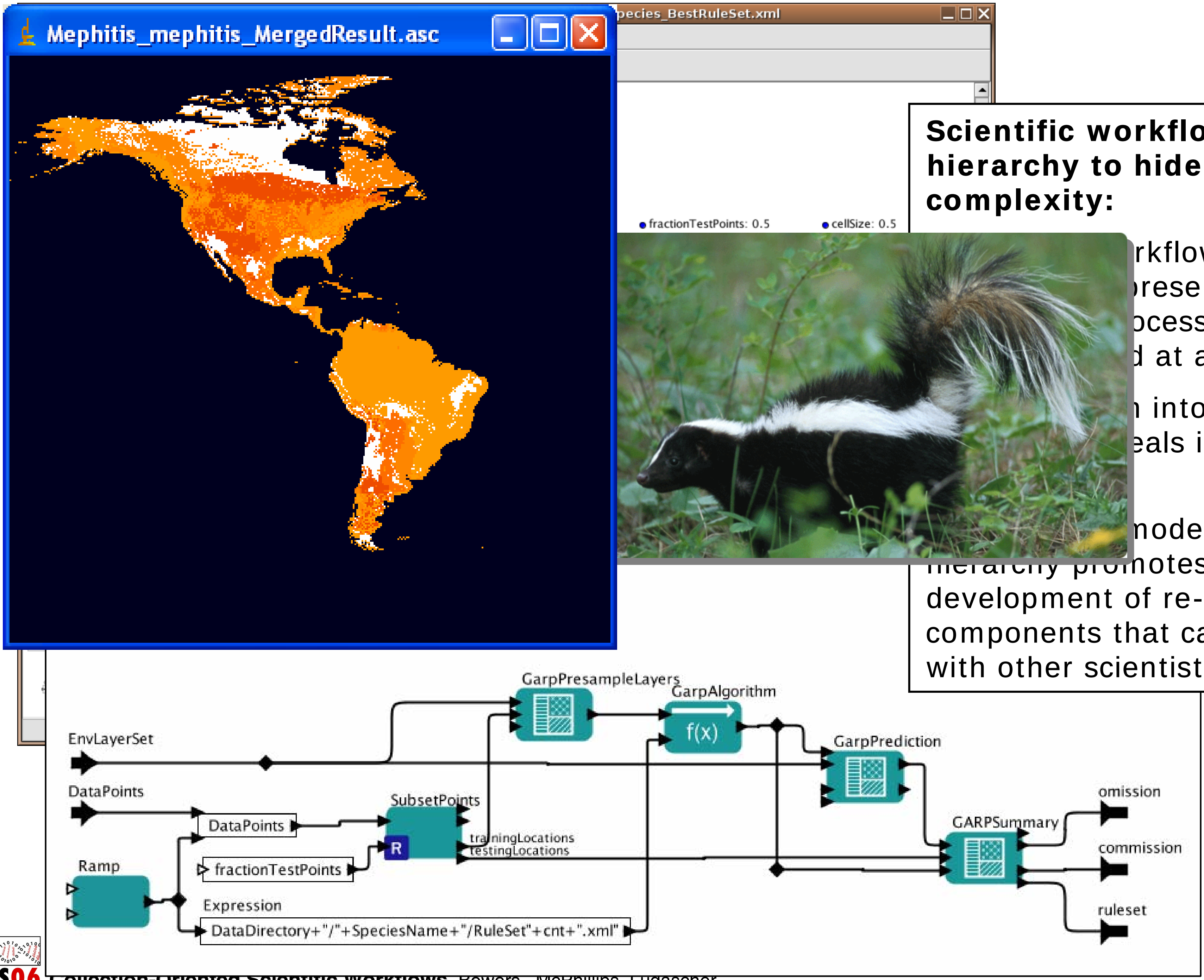
Remote data source



R processing script

```
res ← lm(BARO ~ T_AIR)
res
plot(T_AIR, BARO)
abline(res)
```

Managing complexity: Actor-oriented Modeling & Design



Scientific workflows use hierarchy to hide complexity:

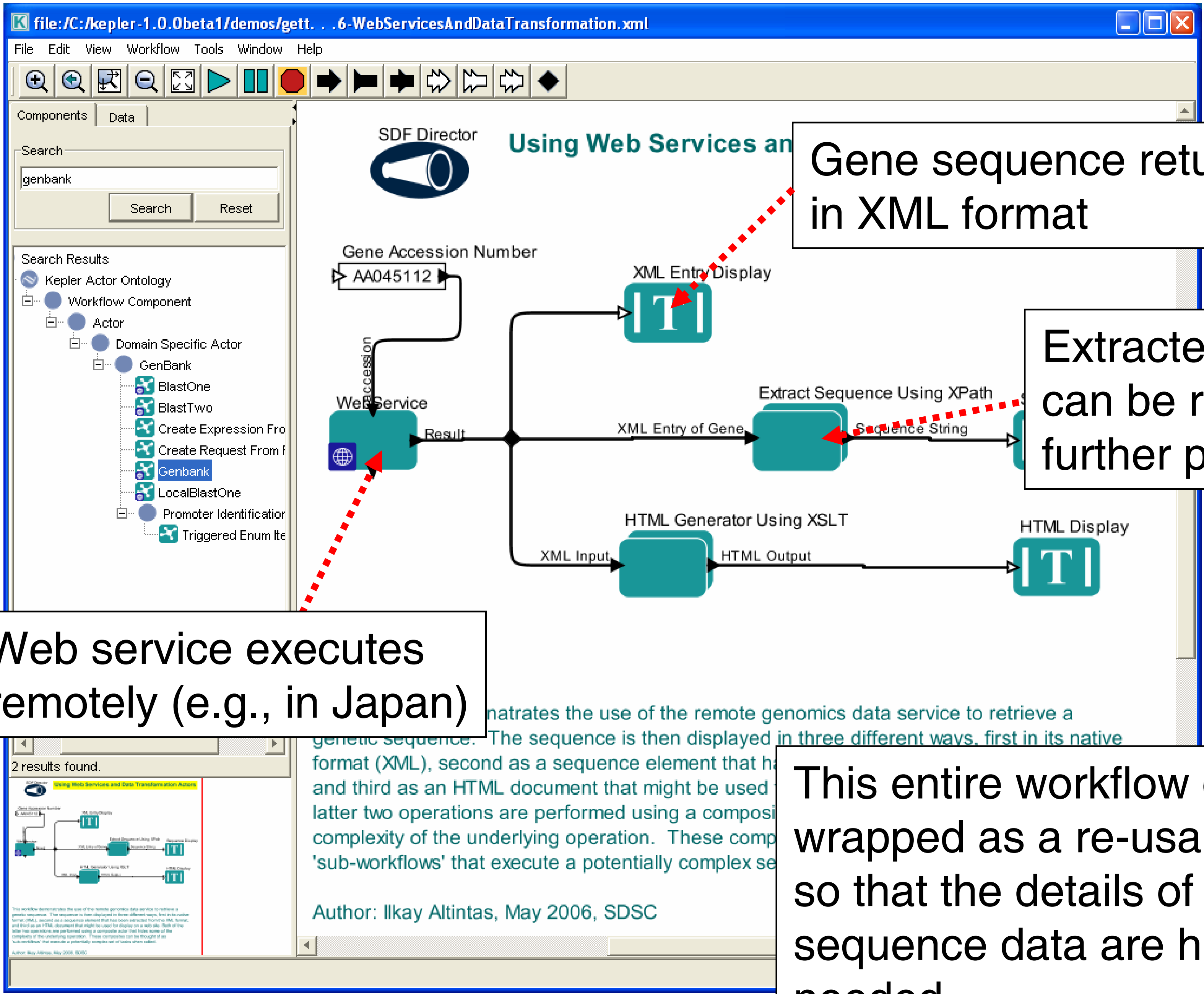
rkflows can be a
presentation of
ocess that is easy
d at a glance

n into sub-
eals increasing

models using
hierarchy promotes the development of re-usable components that can be shared with other scientists

UCDAVIS UNIVERSITY OF CALIFORNIA

Retrieving gene sequences via web services



...natrates the use of the remote genomics data service to retrieve a genetic sequence. The sequence is then displayed in three different ways, first in its native format (XML), second as a sequence element that ha... and third as an HTML document that might be used ... latter two operations are performed using a composi... complexity of the underlying operation. These comp... 'sub-workflows' that execute a potentially complex se...

Author: Ilkay Altintas, May 2006, SDSC

Gene sequence returned in XML format

Extracted sequence can be returned for further processing

Web service executes remotely (e.g., in Japan)

This entire workflow can be wrapped as a re-usable component so that the details of extracting sequence data are hidden unless needed.

Kepler/SEEK Semantic Extensions

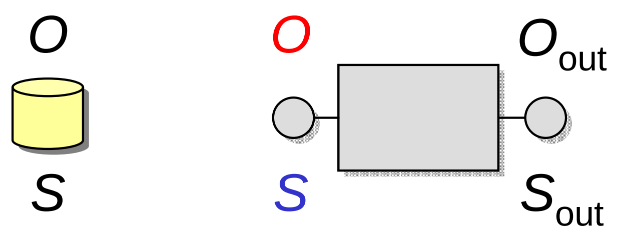
- **Employ semantic extensions (ontologies) for ..**

🟨 data ⇨ 🟩 actor (***data binding***)
🟨 data ⇨ 🟨 data (***data integration / merging / interlinking***)
🟩 actor ⇨ 🟩 actor (***actor / workflow composition***)

  - Smart **S**earch (➔ *Resource Discovery*)
  - Smart **A**ttach (➔ *Data Binding*)
  - Smart **I**ntegration (➔ *Merge Actor*)
  - Smart **L**inks (➔ *Actor Composition*)

*Search-**A**ttach-**I**ntegrate-**L**ink*

“Hybrid” Types … Semantic + Structural Typing



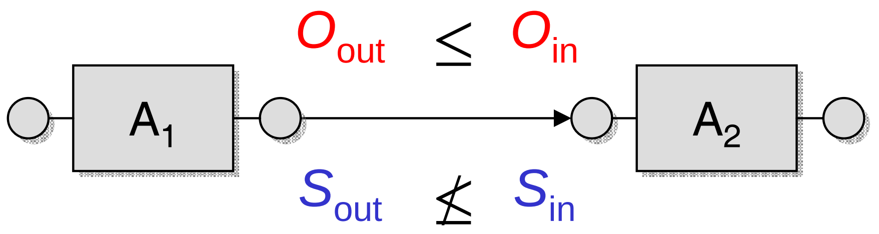
O : Observation $\sqcap$
$\forall$obsProperty.SpeciesOccurrence
S : SpeciesData(site, day, spp, occ)

Structural Types: Given a structural type language $\mathcal{S}$

- Datasets, inputs, and outputs can be assigned structural types $S \in \boldsymbol{S}$

Semantic Types: Given an ontology language $\mathcal{O}$ (e.g., OWL-DL)

- Datasets, inputs, and outputs can be assigned ontology types $O \in \boldsymbol{O}$

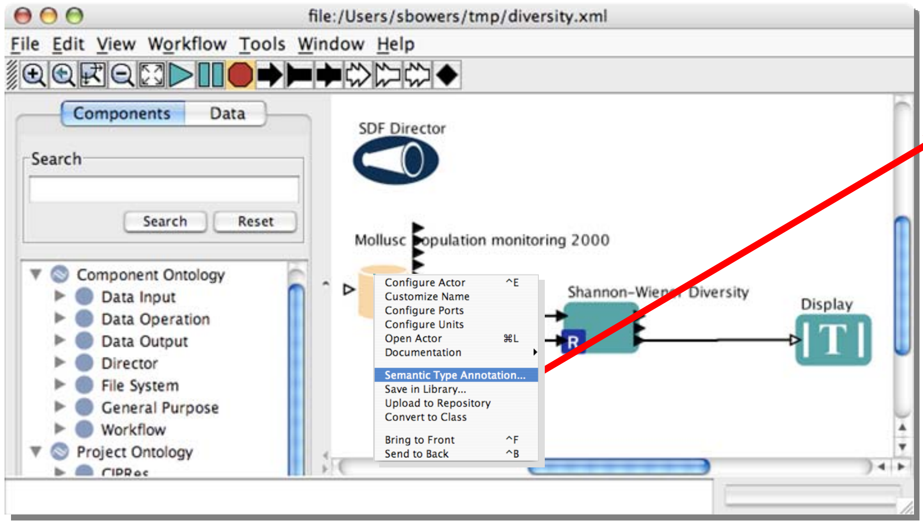


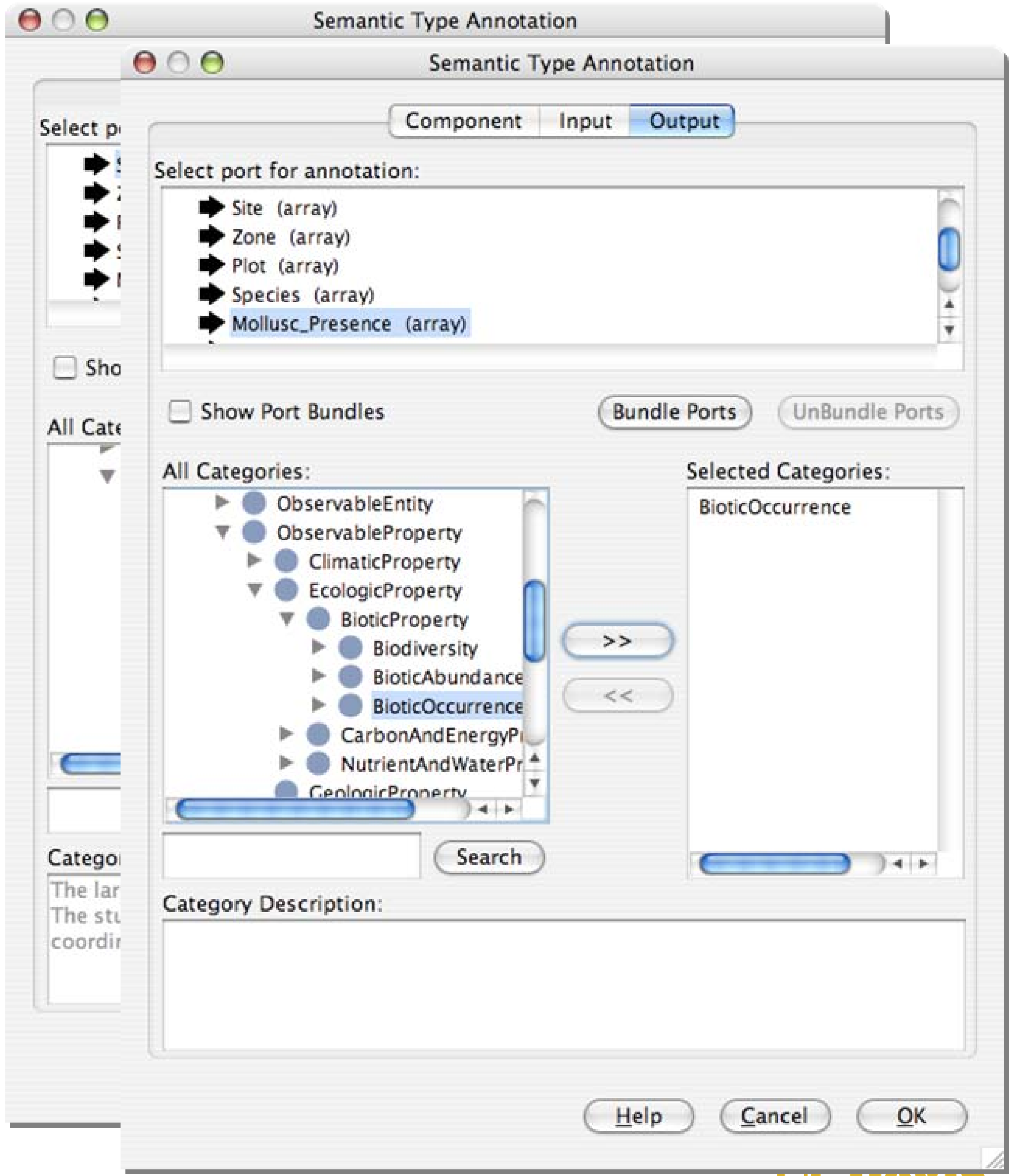
Semantically compatible
but structurally incompatible

Semantic & structural types can be **combined using logic constraints**

$$\alpha := (\forall site,day,sp,occ)\ \text{SpeciesData}(site, day, sp, occ) \rightarrow$$
$$(\exists y)\ \text{Observation}(y), \text{obsProp}(y, occ), \text{SpeciesOccurrence}(occ)$$

Semantic Type Annotation in Kepler





Select port for annotation:
Site (array)
Zone (array)
Plot (array)
Species (array)
Mollusc_Presence (array)

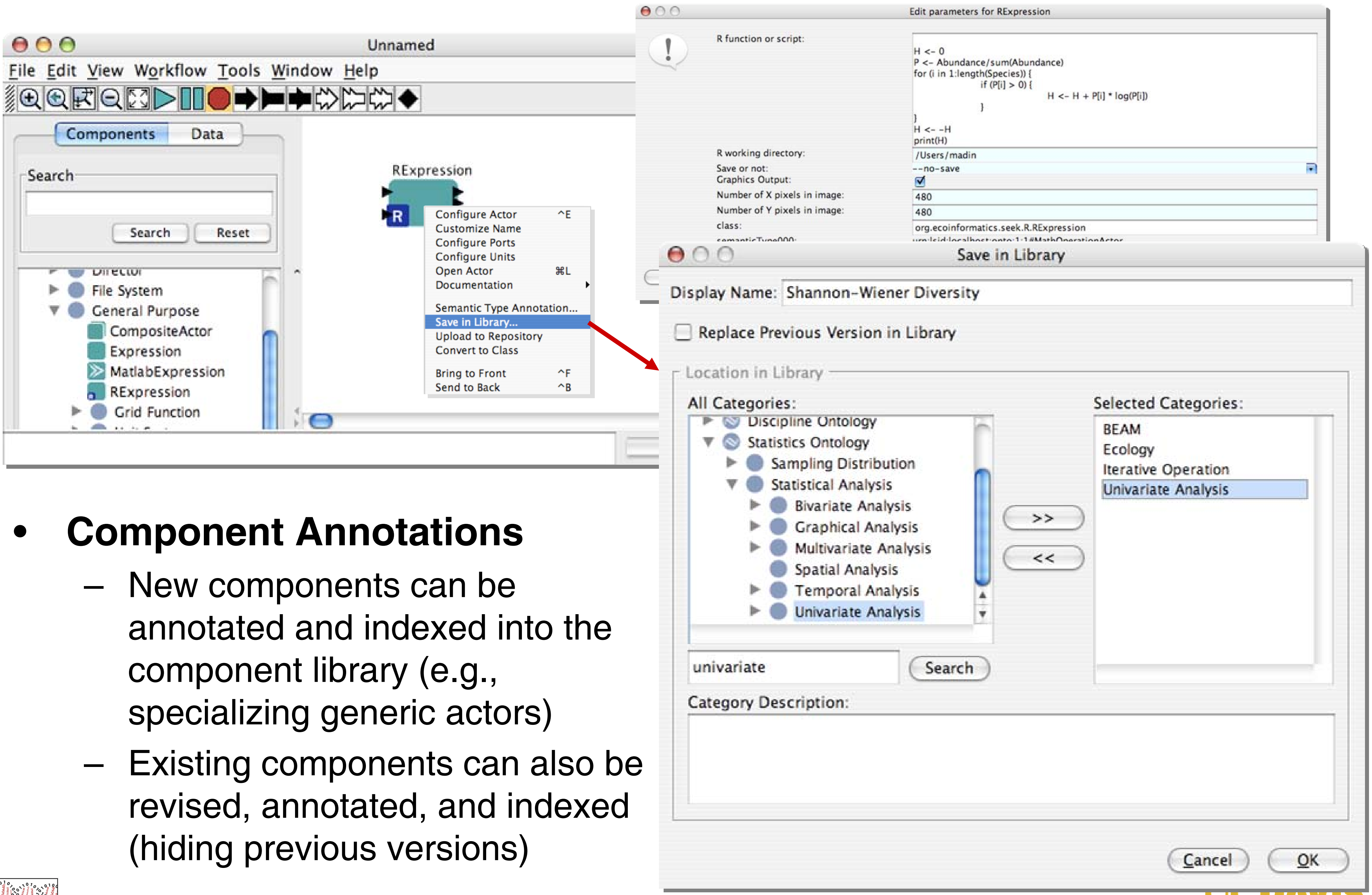
All Categories:
ObservableEntity
ObservableProperty
ClimaticProperty
EcologicProperty
BioticProperty
Biodiversity
BioticAbundance
BioticOccurrence
CarbonAndEnergyP
NutrientAndWaterPr
GeologicProperty

Selected Categories:
BioticOccurrence

Category Description:

- **Component input and output port annotation**
  - Each port can be annotated with multiple classes from multiple ontologies
  - Annotations are stored within the component metadata

Component Annotation and Indexing



R function or script:

```
H <- 0
P <- Abundance/sum(Abundance)
for (i in 1:length(Species)) {
        if (P[i] > 0) {
                H <- H + P[i] * log(P[i])
        }
}
H <- -H
print(H)
```

R working directory:	/Users/madin
Save or not:	--no-save
Graphics Output:	✓
Number of X pixels in image:	480
Number of Y pixels in image:	480
class:	org.ecoinformatics.seek.R.RExpression
semanticType000:	urn:lsid:localhost:onto:1:1#MathOperationActor

Display Name: Shannon-Wiener Diversity

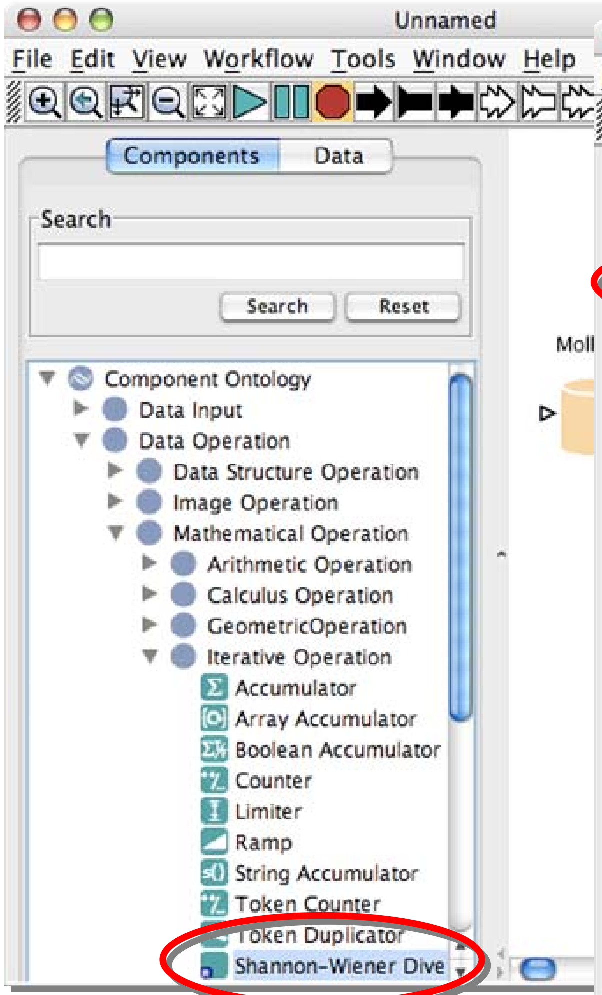
☐ Replace Previous Version in Library

Selected Categories: BEAM, Ecology, Iterative Operation, Univariate Analysis

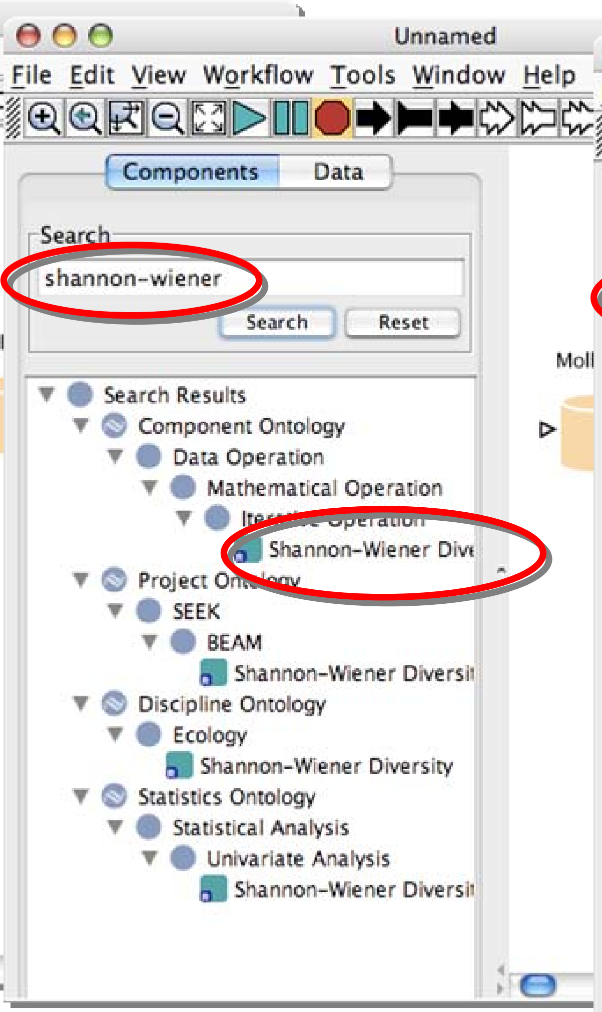
Search: univariate

- **Component Annotations**
  - New components can be annotated and indexed into the component library (e.g., specializing generic actors)
  - Existing components can also be revised, annotated, and indexed (hiding previous versions)

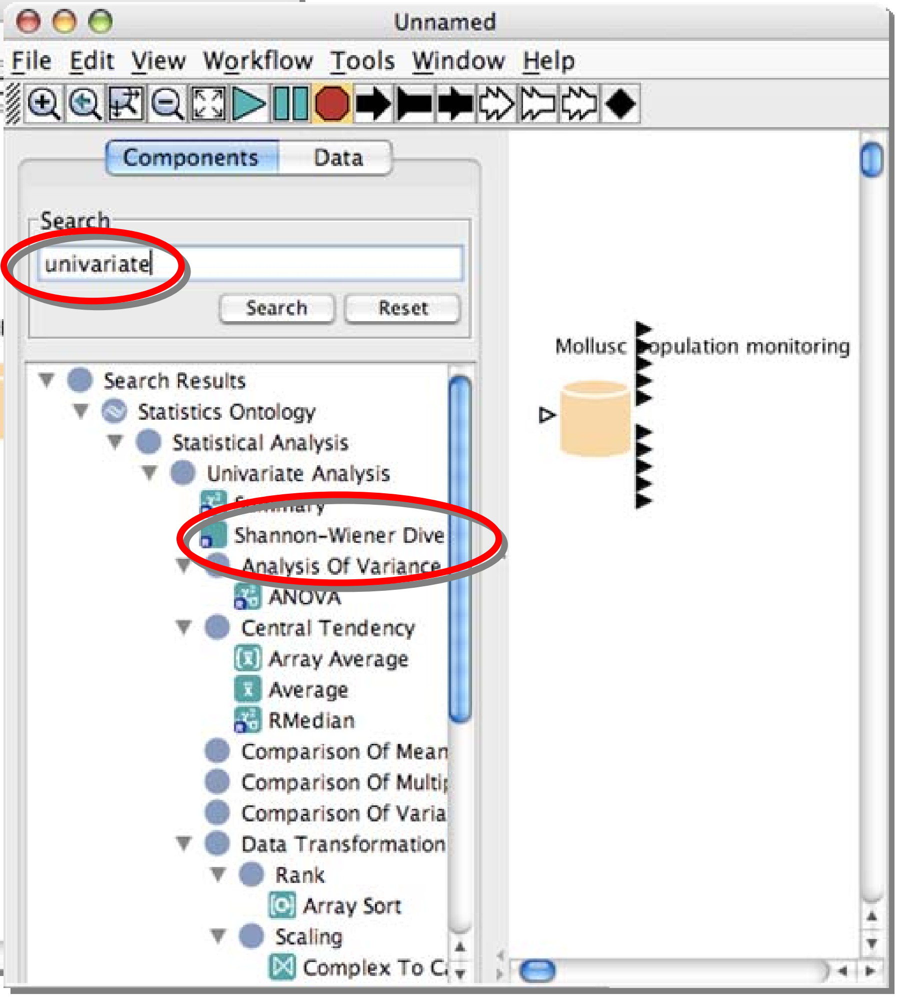
Smart Search



Browse for Components



Search for Component Name

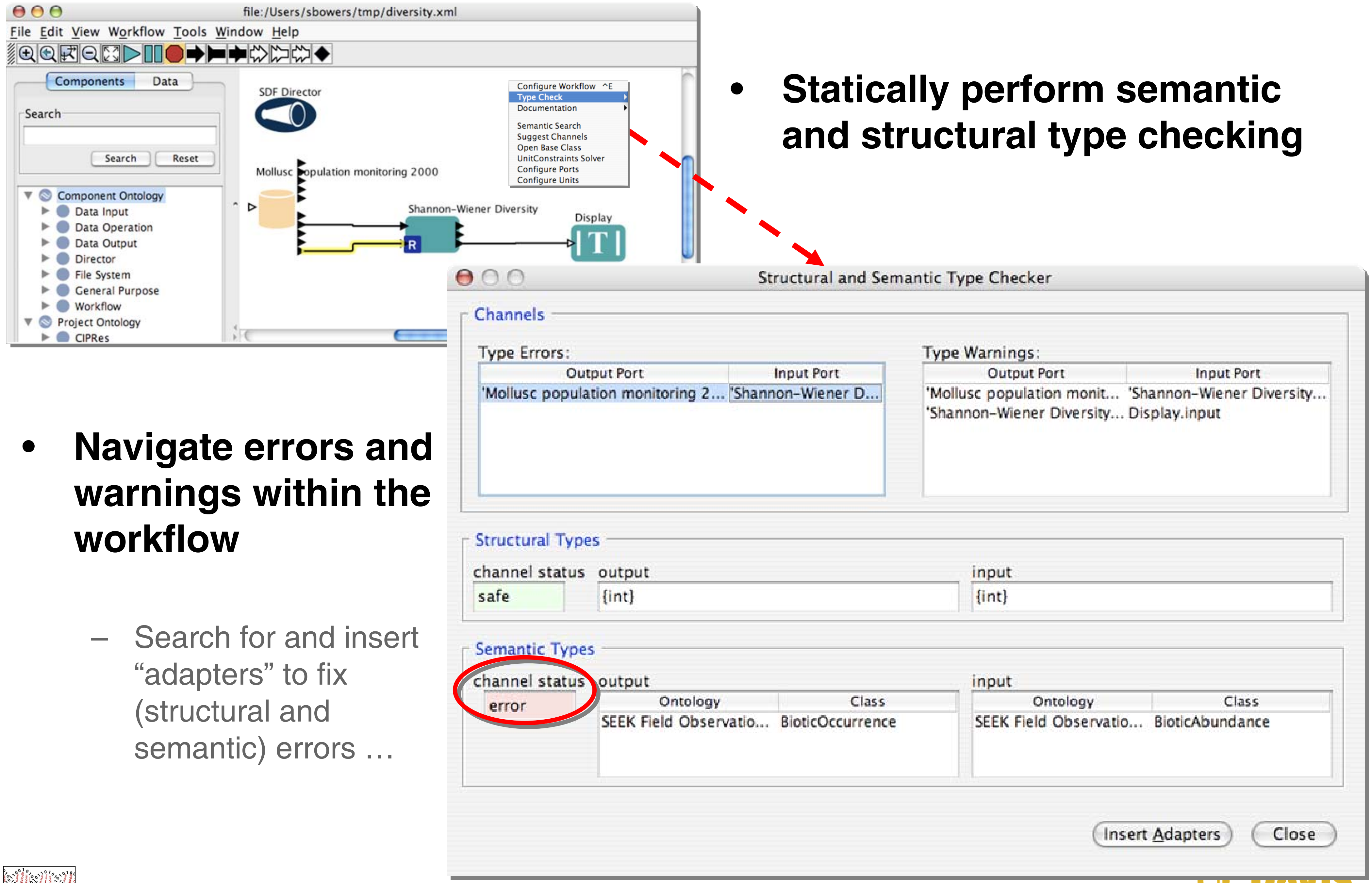


Search for Category / Keyword

Find a component (here: an actor) **in different locations ("categories")**

- … based on the **semantic annotation** of the component (or its ports)
- … registered to one or more ontologies (controlled vocabularies)

Smart Linking (Workflow Design)



Type Errors:

Output Port	Input Port
'Mollusc population monitoring 2...	'Shannon-Wiener D...

Type Warnings:

Output Port	Input Port
'Mollusc population monit...	'Shannon-Wiener Diversity...
'Shannon-Wiener Diversity...	Display.input

Structural Types

channel status	output	input
safe	{int}	{int}

Semantic Types

channel status: error

output

Ontology	Class
SEEK Field Observatio...	BioticOccurrence

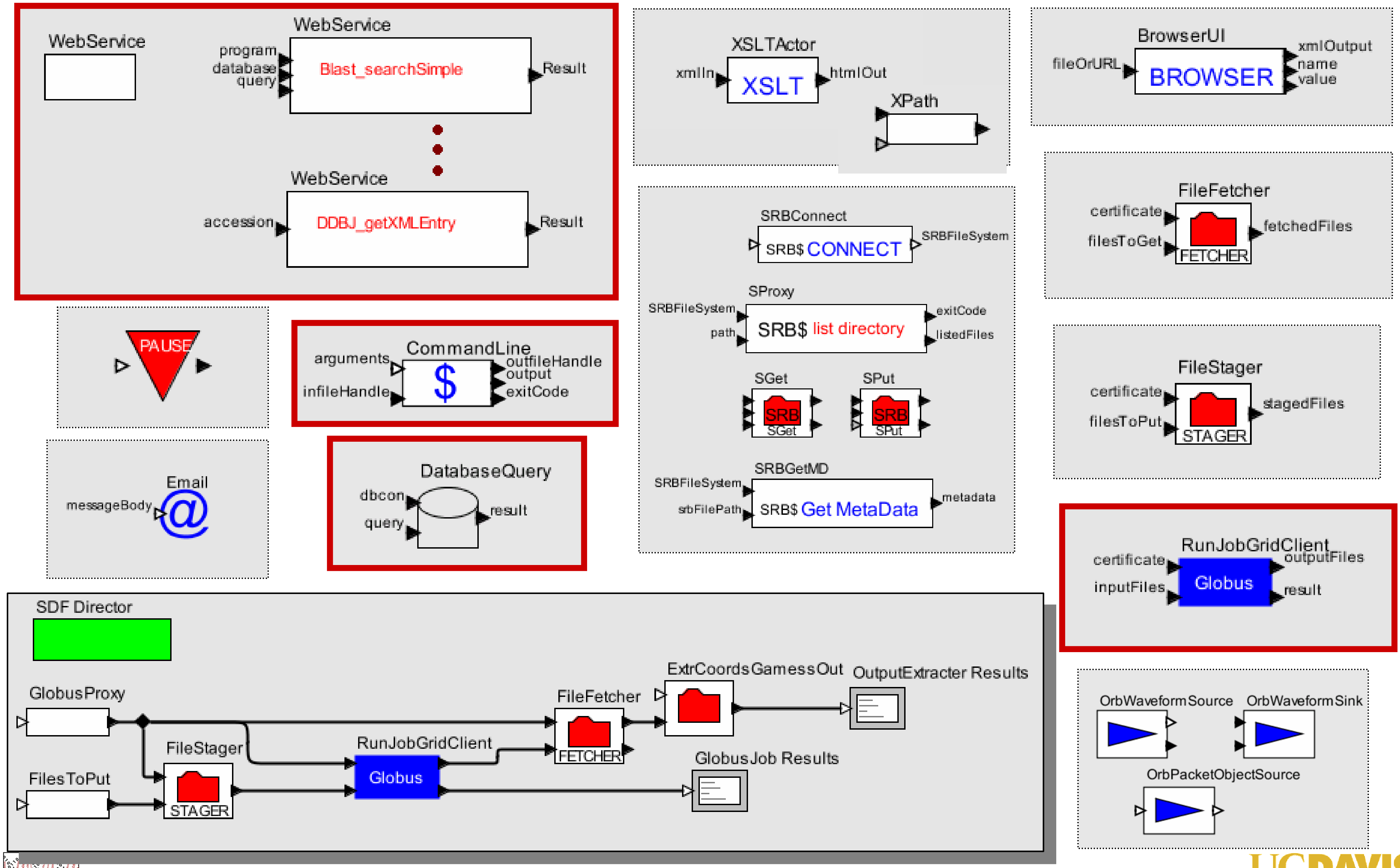
input

Ontology	Class
SEEK Field Observatio...	BioticAbundance

- **Statically perform semantic and structural type checking**

- **Navigate errors and warnings within the workflow**
  - Search for and insert “adapters” to fix (structural and semantic) errors …

Some KEPLER Actors … (oh, the good old days …)



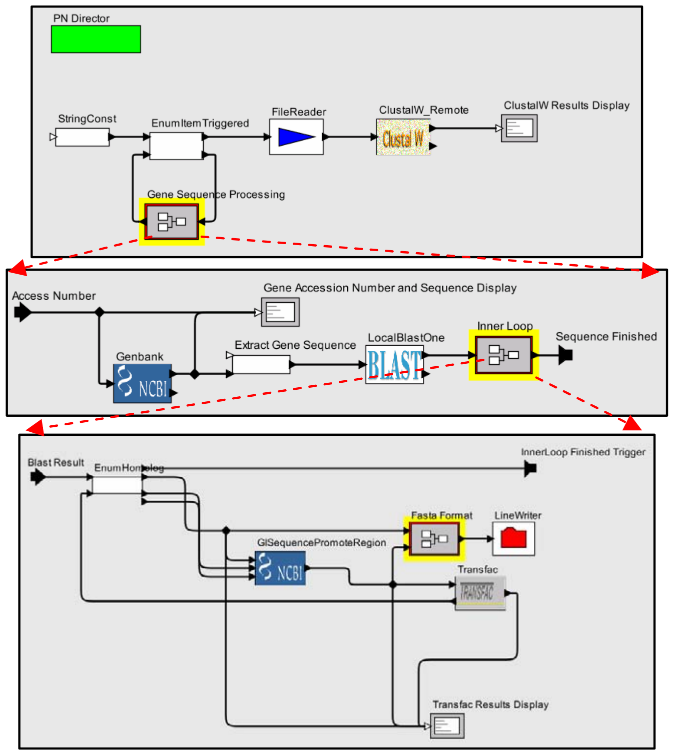
So ...

- **… a question: If scientific workflows are so great, why haven't they taken over the world??**
  - A1: just wait …
  - A2: they already have …
  - A3: The problem of creating flexible, reusable, comprehensible, efficient, … workflows
  - … is akin to the problem of creating modular, reusable, maintainable, … software!
  - … it's complex systems engineering (as in: **difficult**)
    - … and using UML, XML, WS-foo, SOA-bar, and BPEL-baz are no substitute for solving your design problem!

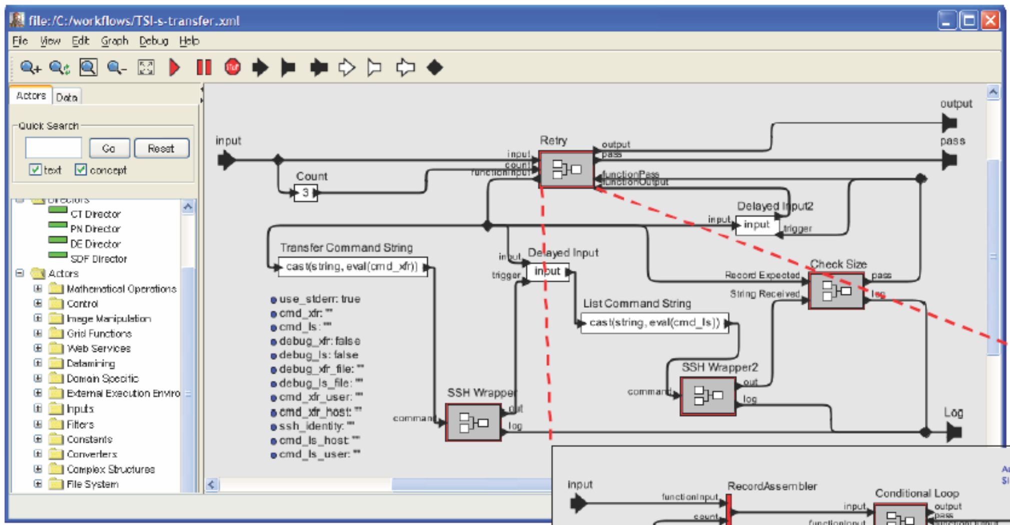
Complexity in Scientific Workflow Design

The use of "control-flow" primitives

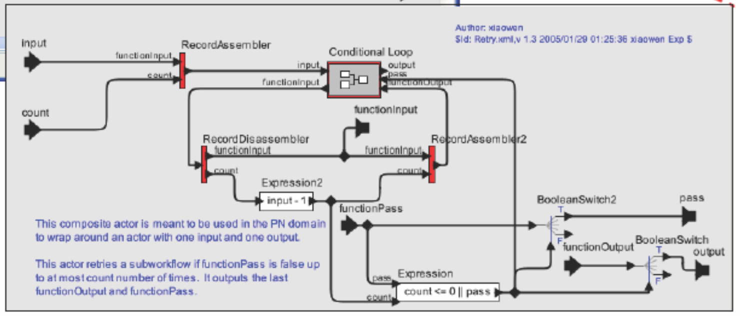
- Managing complex, **nested** data structures (select/filter/transform)
- Fault-tolerance and exception handling
- …



Custom actors, ***hand-crafted*** *control flow limited to sequential execution (SSDBM'03)*



use_stderr: true
cmd_xfr: ""
cmd_ls: ""
debug_xfr: false
debug_ls: false
debug_xfr_file: ""
debug_ls_file: ""
cmd_xfr_user: ""
cmd_xfr_host: ""
ssh_identity: ""
cmd_ls_host: ""
cmd_ls_user: ""



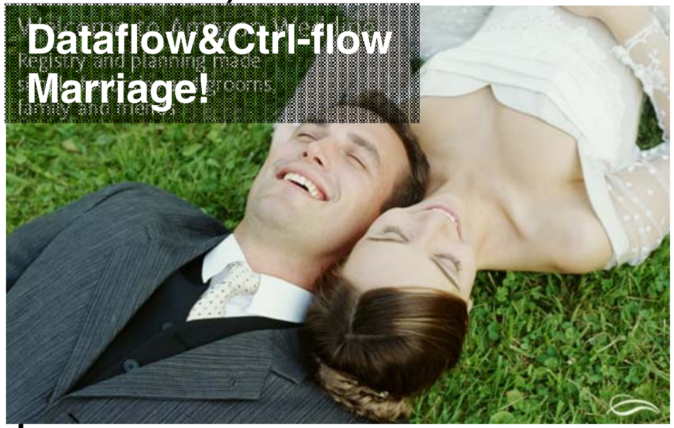
This composite actor is meant to be used in the PN domain to wrap around an actor with one input and one output.

This actor retries a subworkflow if functionPass is false up to at most count number of times. It outputs the last functionOutput and functionPass.

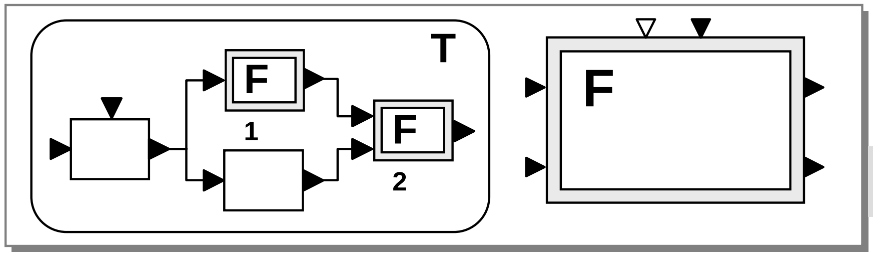
Fault-tolerance control-flow ***"wired-in",*** *e.g. via Boolean switches,* ***complex branching and looping***

Modeling Control-Flow Constructs in Dataflow

- **Dataflow** in Kepler
  - Based on **dataflow process networks** (Kahn et al, Lee et al)
  - Supports **pipeline parallelism** (streaming data)
  - Natural paradigm for **data-driven** workflows
  - Efficient analysis and scheduling
  - **Intuitive model** for workflow designers



- **Control-Flow** in Kepler
  - Branching via if-then-else and switch-case statements
  - Iteration with multiple entry and exit points
  - Low-level actors for manipulating structure (e.g., record-to-array)
- **Problems modeling Control-Flow directly using Dataflow**
  - Overly complicated workflows; hard to understand (low-level programming), maintain, debug, extend ➔ limited reusability; complex re-configuration



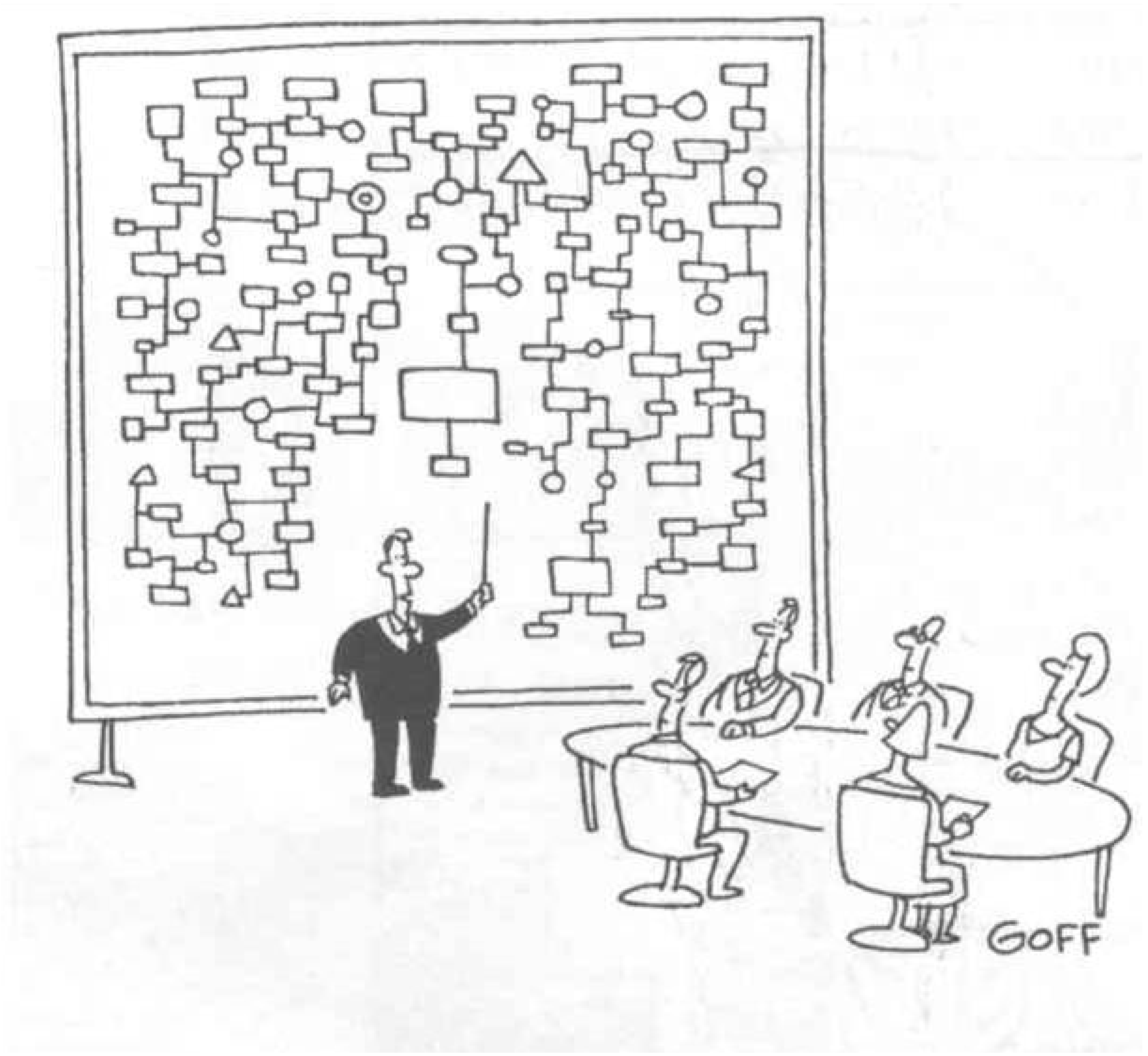
Note to self: Check out the new Taverna control-flow layered architecture !

[Bowers-*et-al*, SciFlow'06]

New Design Primitives (e.g., templates & frames)

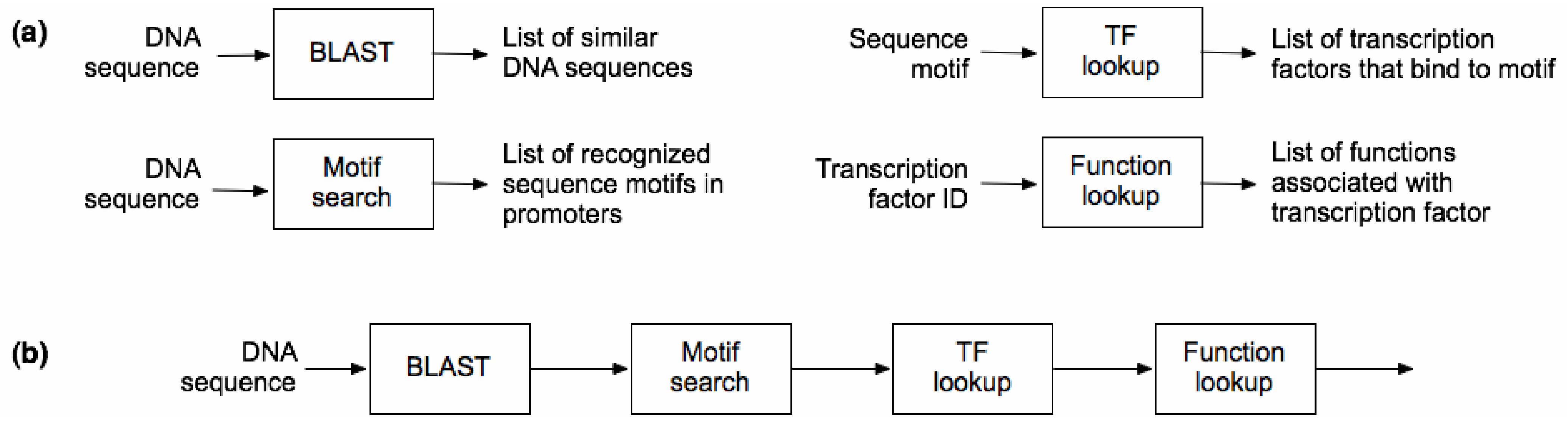
UCDAVIS UNIVERSITY OF CALIFORNIA

Scientific Workflow Design: Challenges



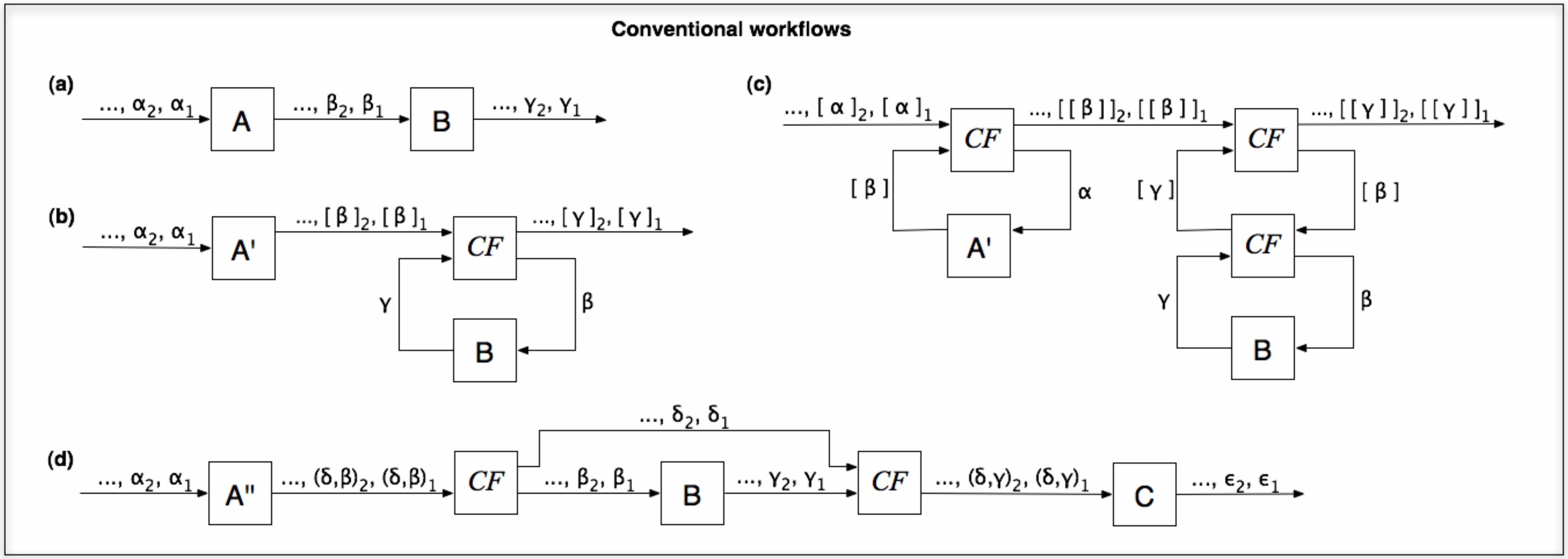
"And that's why our scientific workflows are much **easier to develop, understand and maintain!**"

A Simple Motivating Example


(a)
DNA sequence
BLAST
List of similar DNA sequences
Sequence motif
TF lookup
List of transcription factors that bind to motif
DNA sequence
Motif search
List of recognized sequence motifs in promoters
Transcription factor ID
Function lookup
List of functions associated with transcription factor
(b)
DNA sequence
BLAST
Motif search
TF lookup
Function lookup

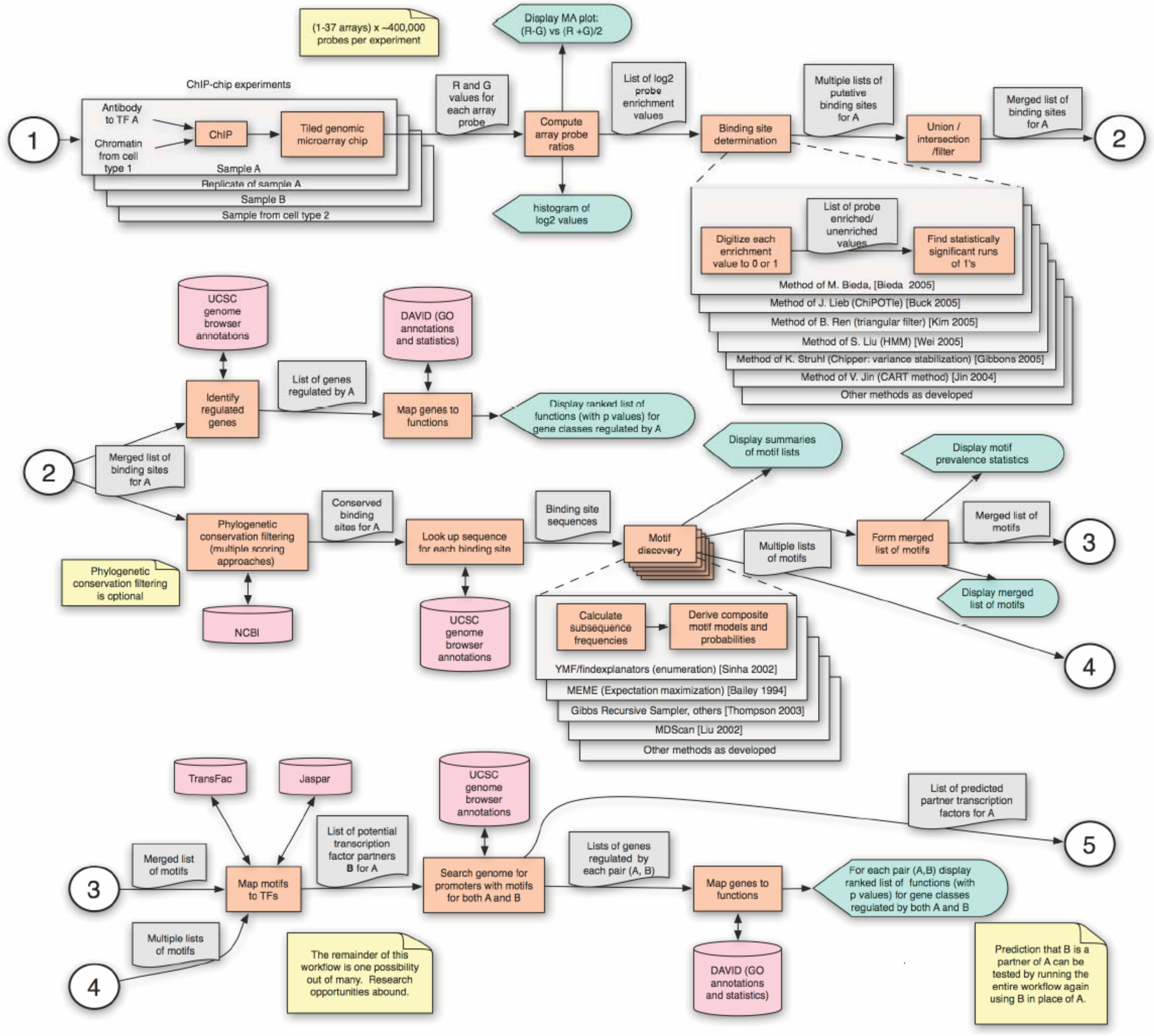
- **Take the services (actors, components) in (a)**
- **… and chain them together in a scientist friendly form a la (b)**
- **… considering the following signatures (cf. Haskell, ML, …)**
  - (c) BLAST :: DNA→ [DNA]
  - (d) MotifSearch :: DNA → [Motif]
  - (e) BLAST o MotifSearch = \x. BLAST(MotifSearch)(x)
- **… oops: (e) is not type correct: note the signatures of (c) and (d)!**
- **a neat solution: implicit or explicit iteration / map(f)[x1,…,xn]**
  - cf. Kepler and Taverna, Kepler solutions

Extended Example: Workflow Evolution



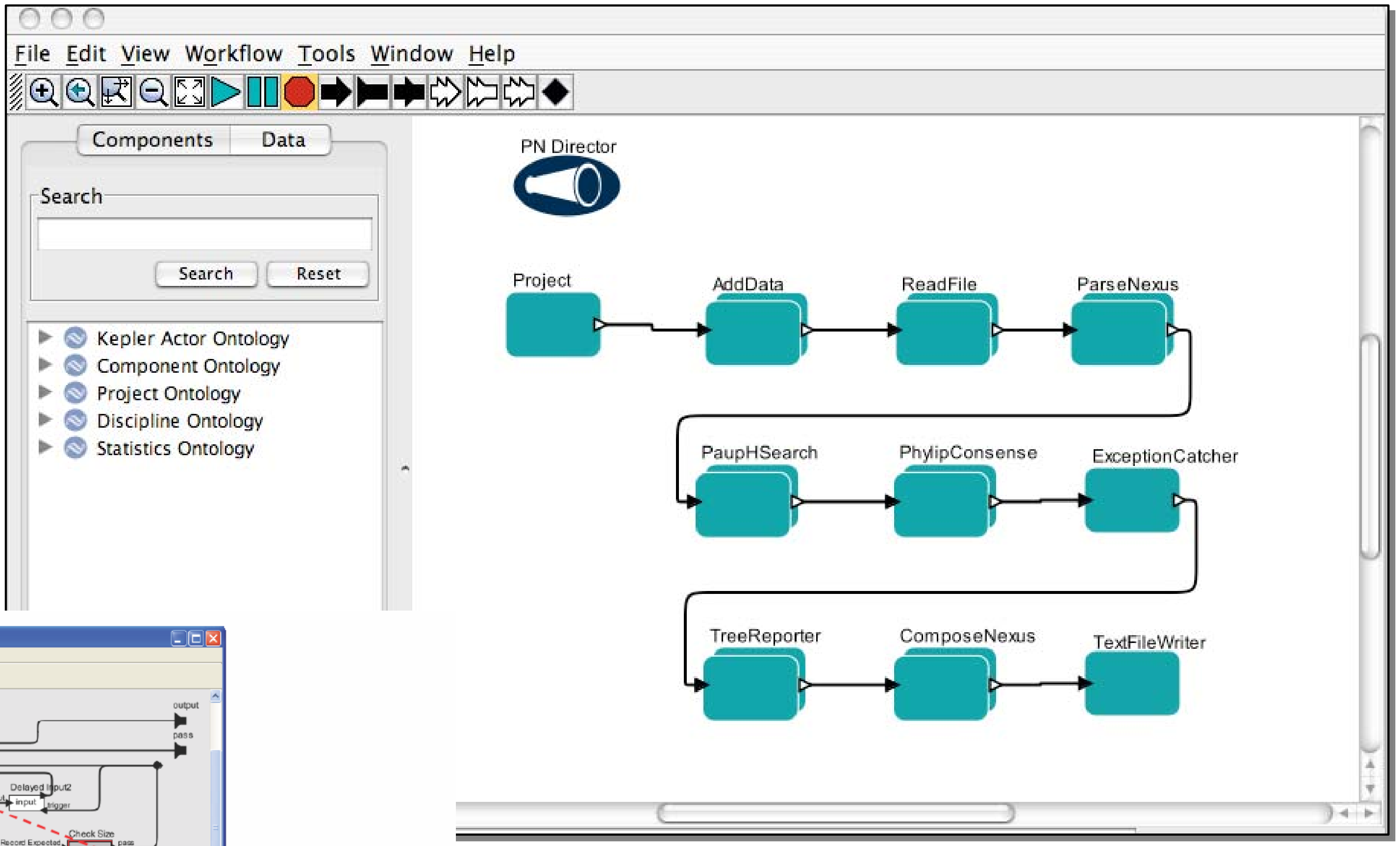
- **(a) => (b): replace *A::a→b* with *A'::a→[b]***
  - need to call B iteratively i.e. wrap B inside a component or add control-flow
- **(b) => (c): upstream produces *[a], [a], …* instead of *a, a, …***
- **(d): need to "bypass" data components since *B* can't handle *d*s**
- **This gets messy quickly …**

A Realistic Example (ChIP-chip workflow)

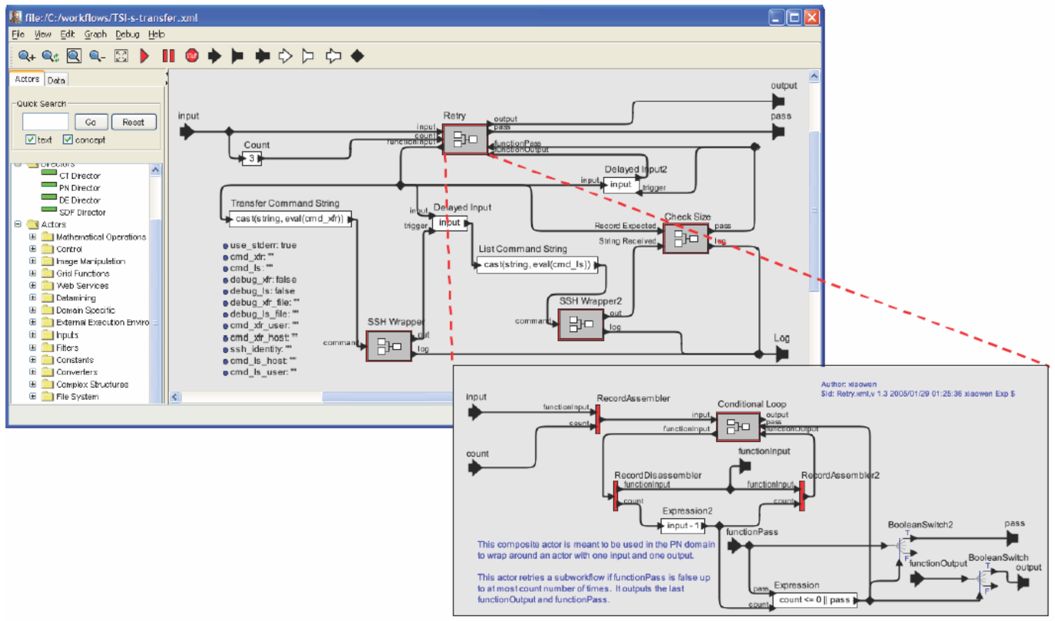


(1-37 arrays) x ~400,000 probes per experiment
Display MA plot: (R-G) vs (R +G)/2
ChIP-chip experiments
1
Antibody to TF A
Chromatin from cell type 1
ChIP
Tiled genomic microarray chip
Sample A
Replicate of sample A
Sample B
Sample from cell type 2
R and G values for each array probe
Compute array probe ratios
histogram of log2 values
List of log2 probe enrichment values
Binding site determination
Multiple lists of putative binding sites for A
Union / intersection /filter
Merged list of binding sites for A
2
Digitize each enrichment value to 0 or 1
List of probe enriched/ unenriched values
Find statistically significant runs of 1's
Method of M. Bieda, [Bieda 2005]
Method of J. Lieb (ChIPOTle) [Buck 2005]
Method of B. Ren (triangular filter) [Kim 2005]
Method of S. Liu (HMM) [Wei 2005]
Method of K. Struhl (Chipper: variance stabilization) [Gibbons 2005]
Method of V. Jin (CART method) [Jin 2004]
Other methods as developed
UCSC genome browser annotations
DAVID (GO annotations and statistics)
Identify regulated genes
List of genes regulated by A
Map genes to functions
Display ranked list of functions (with p values) for gene classes regulated by A
2
Merged list of binding sites for A
Phylogenetic conservation filtering (multiple scoring approaches)
Phylogenetic conservation filtering is optional
NCBI
Conserved binding sites for A
Look up sequence for each binding site
UCSC genome browser annotations
Binding site sequences
Motif discovery
Display summaries of motif lists
Multiple lists of motifs
Form merged list of motifs
Display motif prevalence statistics
Merged list of motifs
3
Display merged list of motifs
4
Calculate subsequence frequencies
Derive composite motif models and probabilities
YMF/findexplanators (enumeration) [Sinha 2002]
MEME (Expectation maximization) [Bailey 1994]
Gibbs Recursive Sampler, others [Thompson 2003]
MDScan [Liu 2002]
Other methods as developed
TransFac
Jaspar
UCSC genome browser annotations
List of predicted partner transcription factors for A
5
3
Merged list of motifs
Map motifs to TFs
List of potential transcription factor partners B for A
Search genome for promoters with motifs for both A and B
Lists of genes regulated by each pair (A, B)
Map genes to functions
For each pair (A,B) display ranked list of functions (with p values) for gene classes regulated by both A and B
4
Multiple lists of motifs
The remainder of this workflow is one possibility out of many. Research opportunities abound.
DAVID (GO annotations and statistics)
Prediction that B is a partner of A can be tested by running the entire workflow again using B in place of A.

So how to get from messy to clean & reusable designs?



before…



- use_stderr: true
- cmd_xfr: ""
- cmd_ls: ""
- debug_xfr: false
- debug_ls: false
- debug_xfr_file: ""
- debug_ls_file: ""
- cmd_xfr_user: ""
- cmd_xfr_host: ""
- ssh_identity: ""
- cmd_ls_host: ""
- cmd_ls_user: ""

Author: xiaowen
$Id: Retry.xml,v 1.3 2005/01/29 01:25:36 xiaowen Exp $

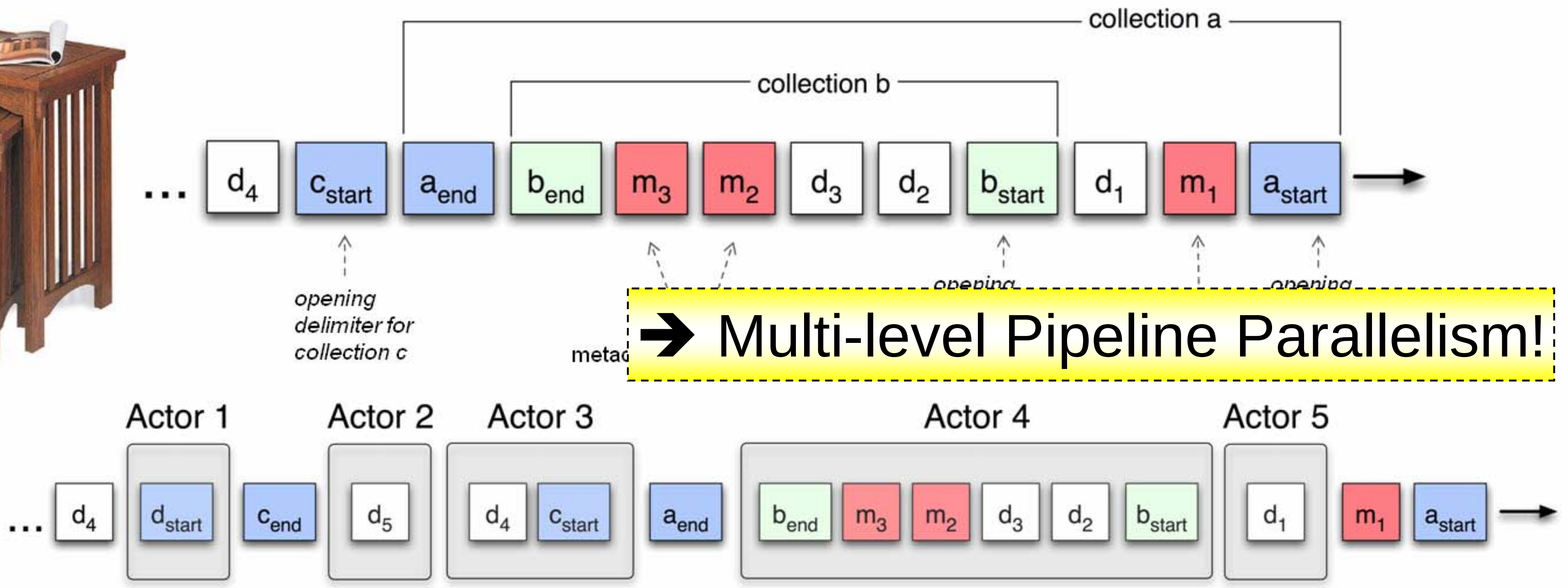
This composite actor is meant to be used in the PN domain to wrap around an actor with one input and one output.

This actor retries a subworkflow if functionPass is false up to at most count number of times. It outputs the last functionOutput and functionPass.

AFTER!

Answer: Collection-Oriented Modeling & Design

- **Collection-Oriented Modeling & Design (COMAD)**
  - starting point: dataflow / actor-oriented modeling & design
  - embrace the **assembly line metaphor fully**
    - ➔ Flow-based Programming (J. Morrison)
  - data = **tagged nested collections**
    - e.g. represented as flattened, **pipelined** (XML) **token streams:**

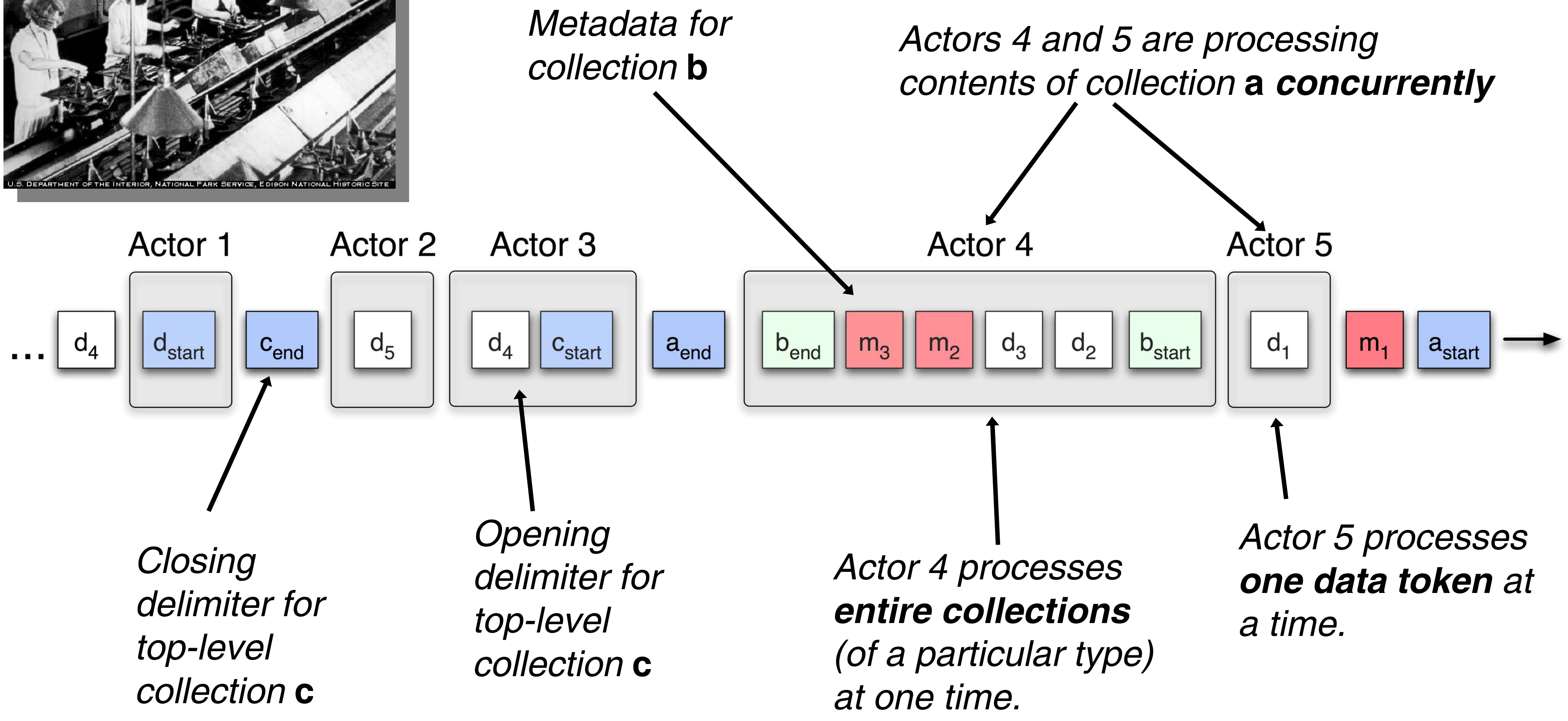


collection a
collection b
... d4 c_start a_end b_end m3 m2 d3 d2 b_start d1 m1 a_start
opening delimiter for collection c
metad
opening
opening
Actor 1
Actor 2
Actor 3
Actor 4
Actor 5
... d4 d_start c_end d5 d4 c_start a_end b_end m3 m2 d3 d2 b_start d1 m1 a_start

➔ Multi-level Pipeline Parallelism!

Multi-level Pipeline Parallelism

U.S. Department of the Interior, National Park Service, Edison National Historic Site

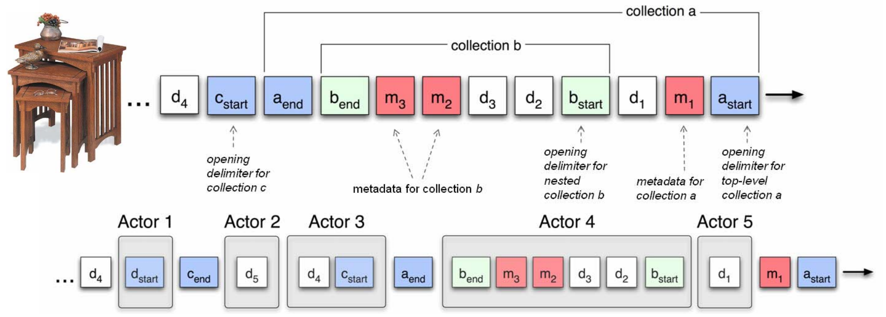

Metadata for collection **b**
Actors 4 and 5 are processing contents of collection **a** ***concurrently***
Actor 1
Actor 2
Actor 3
Actor 4
Actor 5
…
d_4
d_{start}
c_{end}
d_5
d_4
c_{start}
a_{end}
b_{end}
m_3
m_2
d_3
d_2
b_{start}
d_1
m_1
a_{start}
Closing delimiter for top-level collection **c**
Opening delimiter for top-level collection **c**
Actor 4 processes ***entire collections*** (of a particular type) at one time.
Actor 5 processes ***one data token*** at a time.

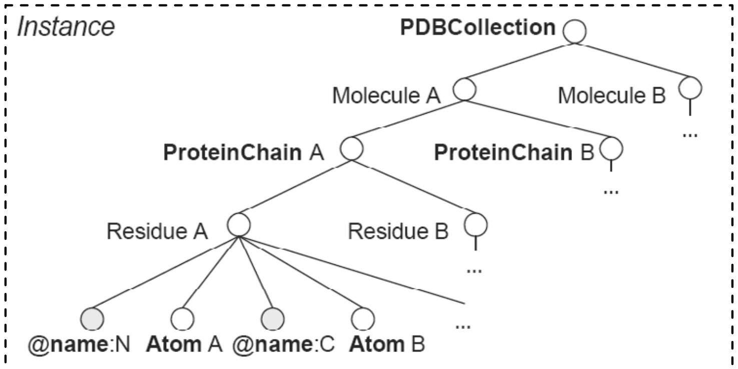
McPhillips,T., Bowers, S., An Approach for Pipelining Nested Collections in Scientific Workflows. *SIGMOD Record*, 34(3), 2005.

How does COMAD work?

- **Some COMAD principles:**
  - data = tagged, flattened, nested collections (token streams)
    - data tokens
    - metadata tokens
      - inherited downwards into (sub)collections
  - define an actor's **read scope** via an (X)Path-like expression:
    - default actor behavior:
      - not mine?
        ➔ don't do anything: **just pass the buck!**
      - stuff within my scope? →
        - » add-only to it (default)
        - » consume scope; write-out result

        (but remember the bypass!)
  - **iteration scope** is a query involving group-by and further refines the granularity/subtrees that constitute the tokens consumed by an actor firing
  - has aspects of implicit iteration (a la Taverna)
    - default iteration level to fix signature mismatches
  - but also:
    - granularity/grouping is definable
    - works on "anything" (assuming scope is matched correctly)
    - add-only and replace modes

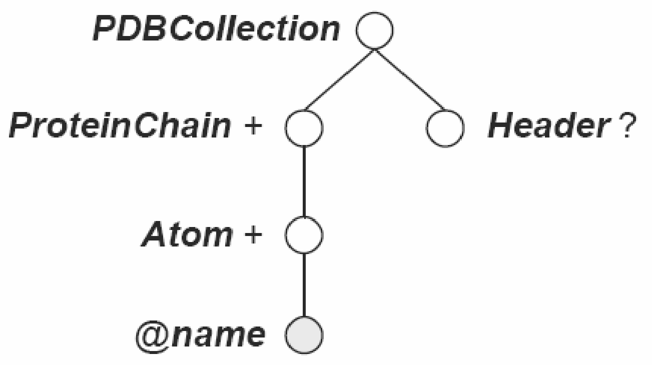

collection a
collection b
… d_4 c_{start} a_{end} b_{end} m_3 m_2 d_3 d_2 b_{start} d_1 m_1 a_{start}
opening delimiter for collection c
metadata for collection b
opening delimiter for nested collection b
metadata for collection a
opening delimiter for top-level collection a
Actor 1
Actor 2
Actor 3
Actor 4
Actor 5
… d_4 d_{start} c_{end} d_5 d_4 c_{start} a_{end} b_{end} m_3 m_2 d_3 d_2 b_{start} d_1 m_1 a_{start}



PDBCollection [
  Molecule [
    ProteinChain [
      Residue [**@name**:N, **Atom**:A, **@name**:C, **Atom**:B, …],
      Residue […]],
    ProteinChain […]],
  Molecule […]]

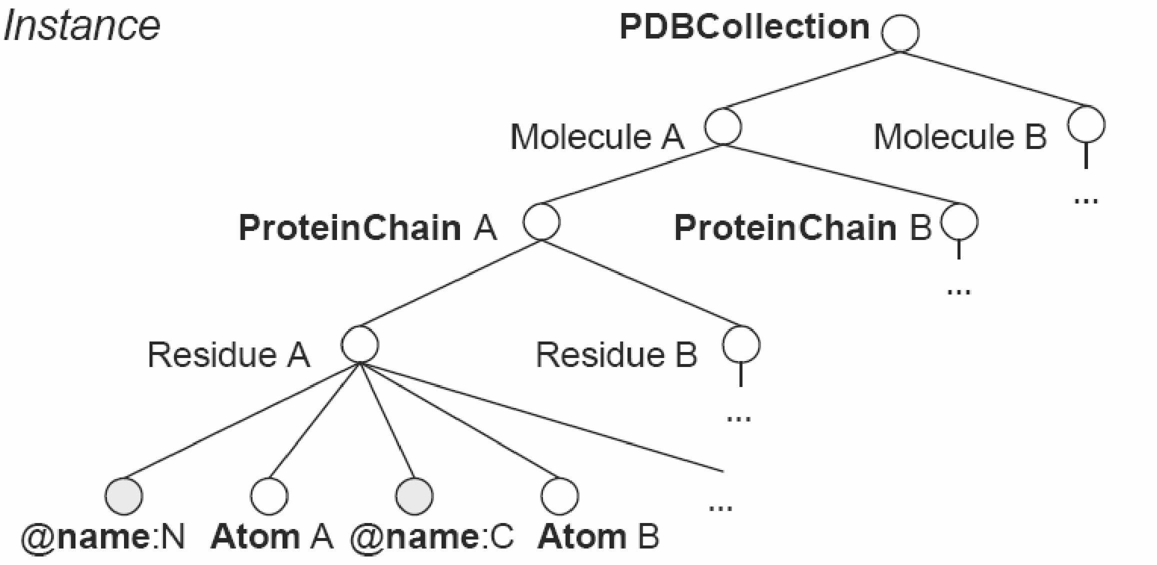
Read Scoping with Collection Schemas (Types)

Schema



```
PDBCollection [
    Header [ ] ?,  ProteinChain [
        Atom with @name + ] +
 ]
```

Instance

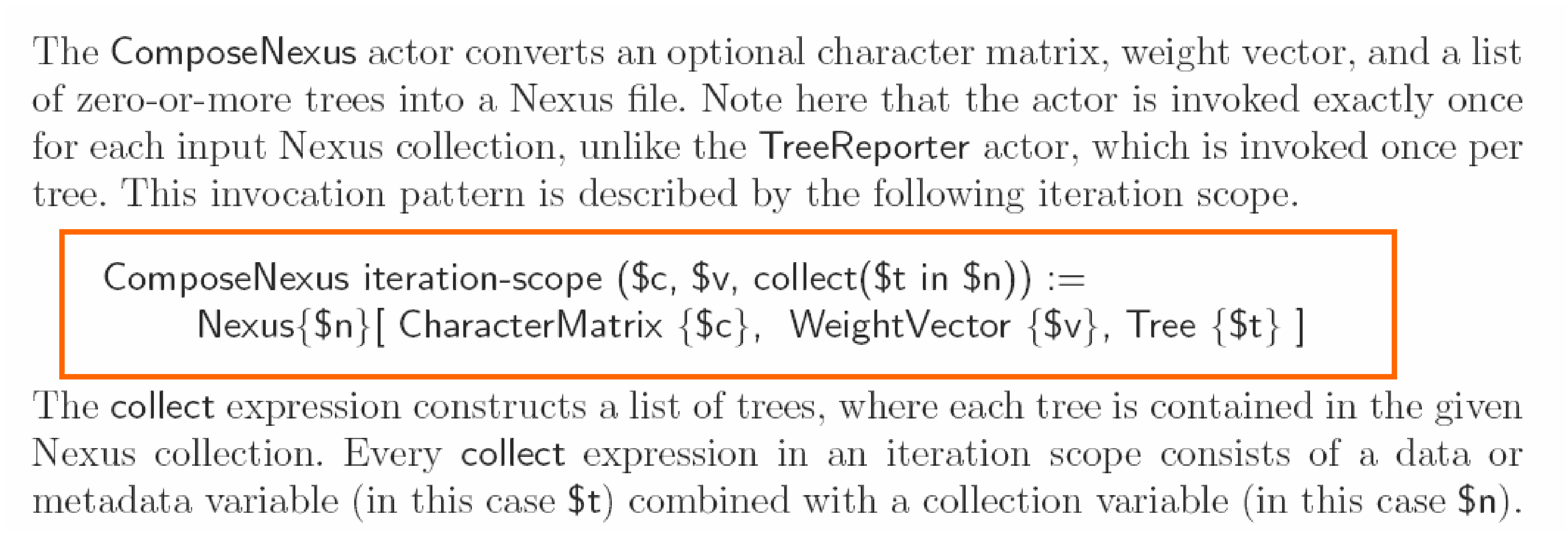


```
PDBCollection [
   Molecule [
      ProteinChain  [
         Residue [ @name:N, Atom:A, @name:C, Atom:B, … ],
         Residue [ …] ],
      ProteinChain […] ],
   Molecule […] ]
```

Collection schema
(defining a query pattern)

… and a sample matching instance

Iteration Scoping via Queries



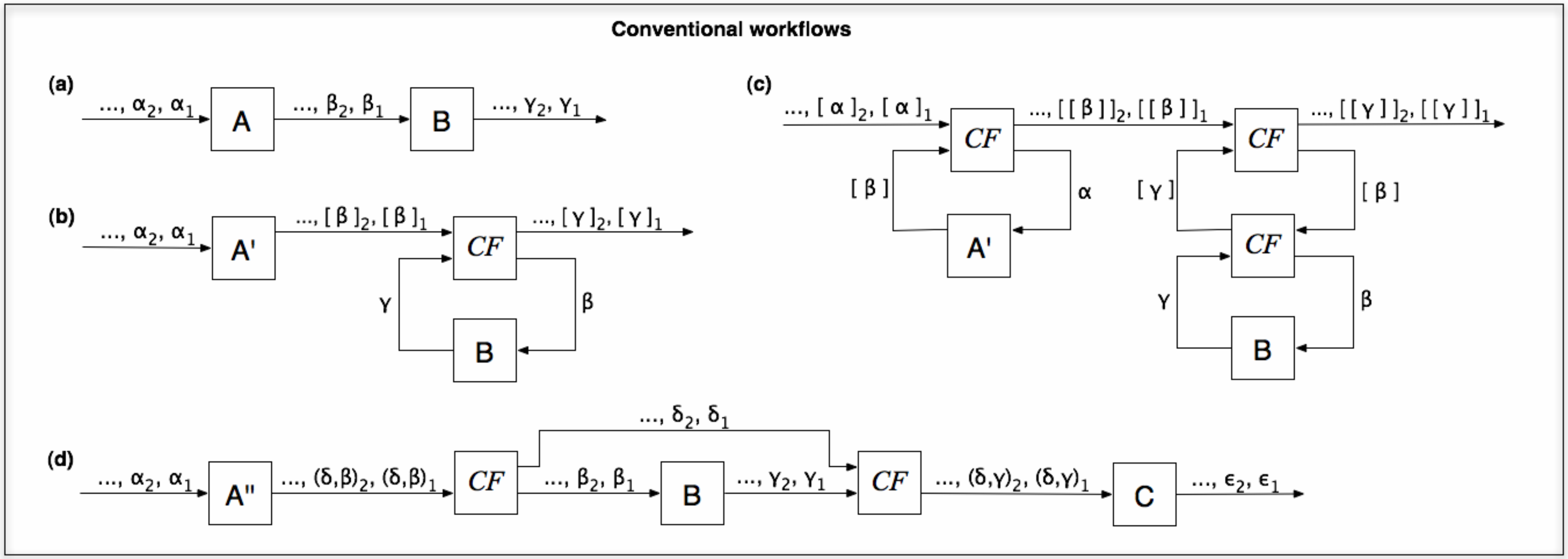
The ComposeNexus actor converts an optional character matrix, weight vector, and a list of zero-or-more trees into a Nexus file. Note here that the actor is invoked exactly once for each input Nexus collection, unlike the TreeReporter actor, which is invoked once per tree. This invocation pattern is described by the following iteration scope.

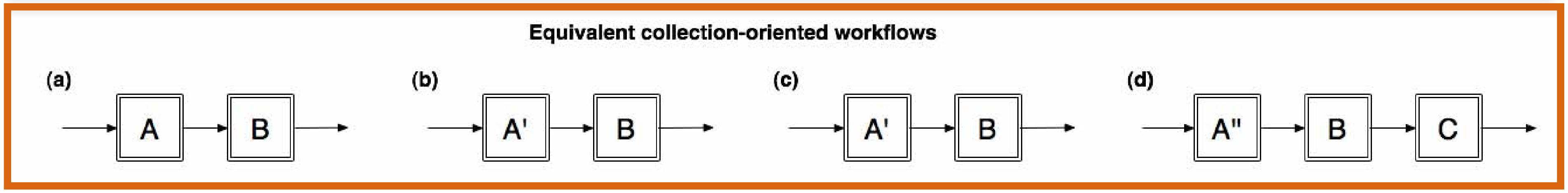
```
ComposeNexus iteration-scope ($c, $v, collect($t in $n)) :=
        Nexus{$n}[ CharacterMatrix {$c},  WeightVector {$v}, Tree {$t} ]
```

The collect expression constructs a list of trees, where each tree is contained in the given Nexus collection. Every collect expression in an iteration scope consists of a data or metadata variable (in this case $t) combined with a collection variable (in this case $n).

- **ingredients / related:**
  - tree pattern queries / XML selection queries
  - … with group-by (here “group-by $c, $v”)
  - see also: list comprehensions as queries (cf. CPL/Biokleisli)

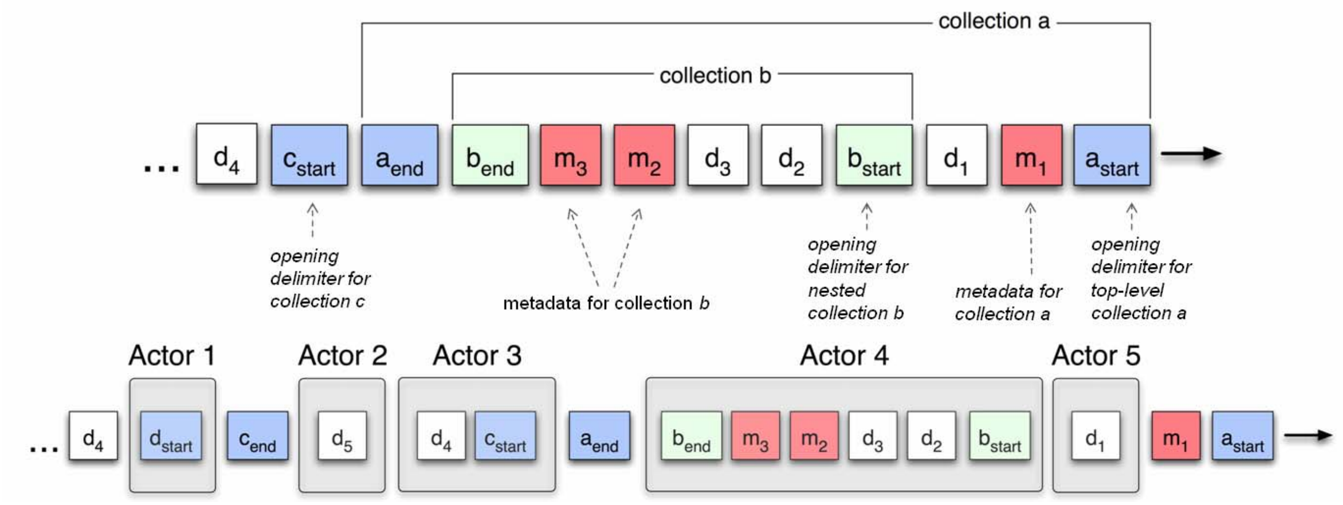
COMAD: What we gained

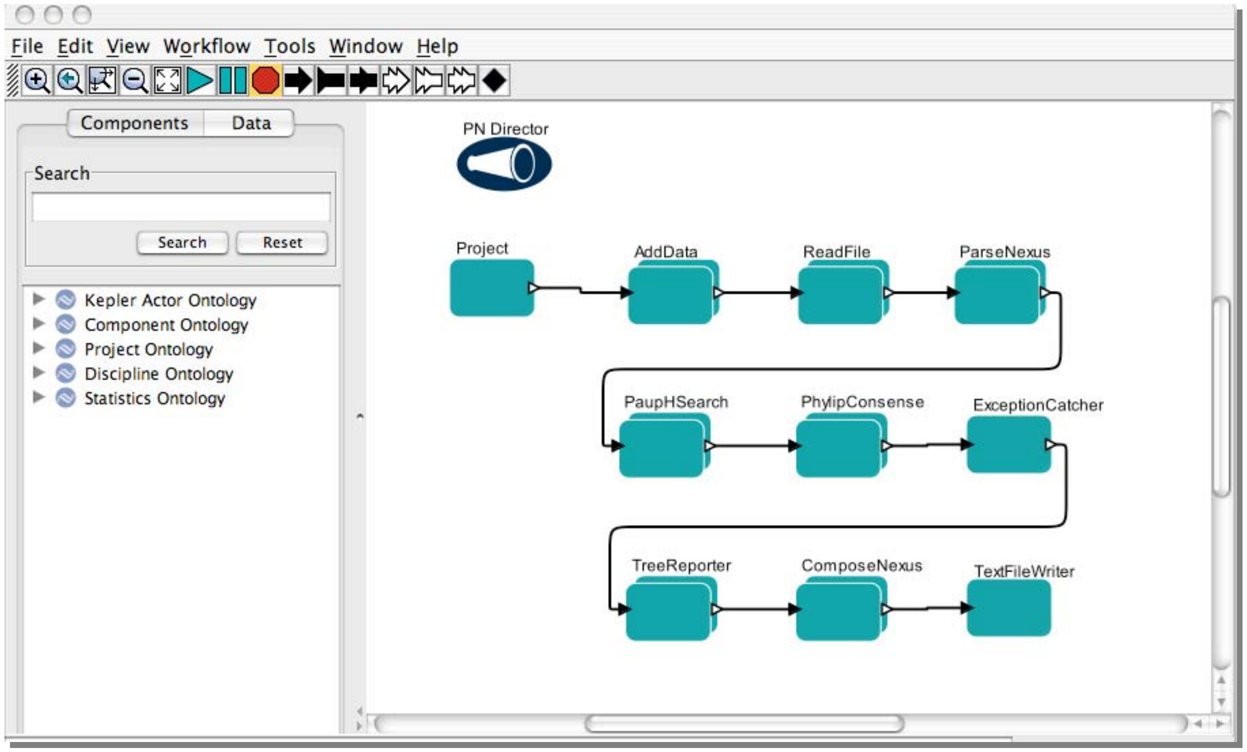




- **from fragile, messy workflow designs**
- **… to more reusable actors**
  - just change the scopes
  - sometimes not even that is needed
- **… and cleaner workflow design**
- **Crux: keep the nesting structure of data (pass through, add-only)**
  … and let it drive the (semi-)implicit iteration

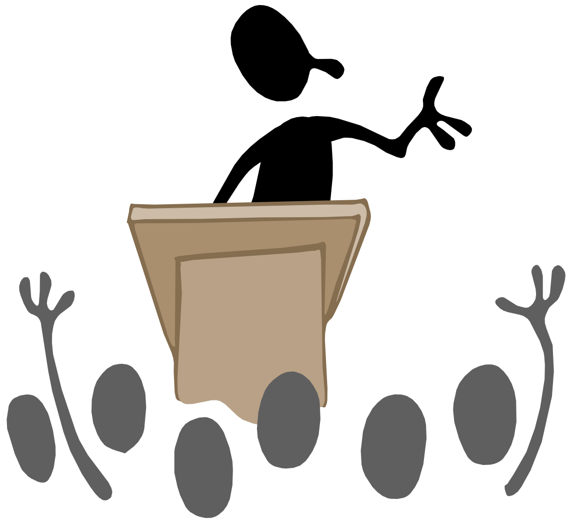
Summary: Collection-Oriented WF Modeling & Design


collection a
collection b
… d4 cstart aend bend m3 m2 d3 d2 bstart d1 m1 astart
opening delimiter for collection c
metadata for collection b
opening delimiter for nested collection b
metadata for collection a
opening delimiter for top-level collection a
Actor 1
Actor 2
Actor 3
Actor 4
Actor 5
… d4 dstart cend d5 d4 cstart aend bend m3 m2 d3 d2 bstart d1 m1 astart



- **Assembly line metaphor: Dataflow + XML …**
  - Streams are nested collections (≈ XML)
  - Less “messy” WFs (more linear, less branching)
  - Pipelined parallelism (stream lists)

Acknowledgements and Q&A ...



- NSF/ITR Science Environment for Ecological Knowledge (SEEK)
- NSF/ITR Geosciences Network (GEON)
- DOE/SciDAC Scientific Data Management Center (SDM)
- U.S. Dept. of Energy, LLNL