

Outline of the Tutorial Modules

I. Overview on Scientific Data Management (Gertz)

13:30–14:15

II. From Conventional to Scientific Data Integration (Ludaescher)

14:15–15:00

III. From Scientific Data Formats to Data Stream Processing (Gertz)

15:30–16:15

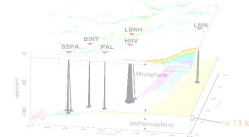
IV. Introduction to Scientific Workflows (Ludaescher)

16:15–17:00

SDM Tutorial, EDBT'06, Gertz, Ludäscher

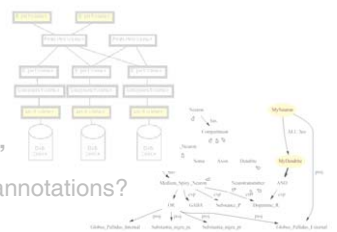
Types of “Integration”

- **Spatial (co-)registration/“overlay” of different data**
 - from 2D, 3D, 4D (x,y,z,t), (4+n) D

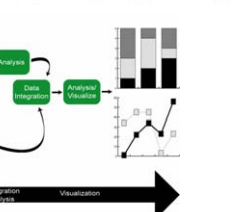


- **Conventional (DB-oriented) integration:**

- schema-based
- view-based
- at the data-level



- **Extended DI approaches using “ontologies”**
 - ontologies? controlled vocabularies? metadata/annotations?



- **Application/process integration**
 - scientific workflows (Module IV)
- **Other mechanisms of “integration”**
 - link-based (Aladin), clustering, ...
 - statistics, data mining, visualization, ...

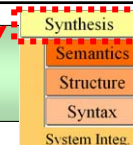
SDM Tutorial, EDBT'06, Gertz, Ludäscher

IV. Introduction to Scientific Workflows (SWF)

- Scientific Workflows in e-Science and CI
- SWF vs Business Workflows
- Features of a SWF System (Kepler)
- Flow-based Programming and Scientific Workflow Design
- Semantic Extensions

SDM Tutorial, EDBT'06, Gertz, Ludäscher

Information Integration Challenges: *S⁵ Heterogeneities*



- **Synthesis** of applications, analysis tools, data & query components, ... into “**scientific workflows**”
 - How to put together components to solve a scientist’s problem?
- **Scientific Problem Solving Environments (PSEs)**
 - **Portals, Workbench (“scientist’s view”)**
 - + ontology-enhanced data registration, discovery, manipulation
 - + creation and registration of new data products from existing ones, ...
 - **Scientific Workflow System (“engineer’s view”)**
 - + for designing, re-engineering, deploying analysis pipelines and scientific workflows; *a tool to make new tools ...*
 - + e.g., creation of new datasets from existing ones, dataset registration, ...

SDM Tutorial, EDBT'06, Gertz, Ludäscher

Motivation: Scientific Workflows, Pre-Cyberinfrastructure

Synthesis
Semantics
Structure
Syntax
System Integ.

- **Data Federation & Grid “Plumbing”:**
 - access, move, replicate, query ... data (**Data-Grid**)
 - authenticate ... SRB Sget/Sput ... OPeNDAP, ... Antelope/ORBs
 - schedule, launch, monitor jobs (**Compute-Grid**)
 - Globus, Condor, Nimrod, APST, ...
- **Data Integration:**
 - Conceptual querying & integration, structure & semantics, e.g. mediation w/ SQL, XQuery + OWL (*Semantics-enabled Mediator*)
- **Data Analysis, Mining, Knowledge Discovery:**
 - manual/textbook (e.g. ternary diagrams), Excel, R, simulations, ...
- **Visualization:**
 - 3-D (volume), 4-D (spatio-temporal), n-D (conceptual views) ...



- **one-of-a-kind custom apps., detached (island) solutions**
- workflows are **hard to reproduce, maintain**
- **no/little** workflow design, automation, reuse, documentation
- need for an **integrated scientific workflow environment**

SDM Tutorial, EDBT'06, Gertz, Ludäscher

Scientific Workflows

A model of the way a scientist works with their data and tools

- Mentally coordinate data export, import, analysis via software systems

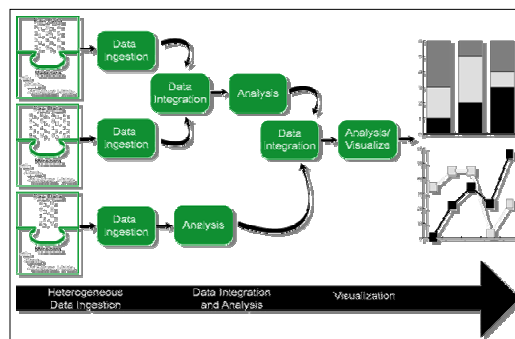
Emphasize dataflow (≠ business workflows)

Metadata: automatic data ingestion, analysis, provenance tracking ...

Goals:

- SWF **automation**
- SWF **component reuse**
- SWF **design & documentation**

... make scientific data analysis and management tasks easier for the scientist!



SDM Tutorial, EDBT'06, Gertz, Ludäscher

Types of Scientific Workflows (overlapping)

- What do we use scientific workflow systems for?

- Short answer:
- ... nearly everything ...



- Types of Scientific Workflows

- **“Modeling & Design”**: Capture or reverse-engineer processes and information flows at all levels
- **“Knowledge discovery”**: Automate repetitive data access, retrieval, custom analysis (e.g. Blast), generic steps (PCA, cluster analysis, ..), Ex: PIW, Motif analysis, NDDP, ...
- **“Plumbing”**: Stage files, submit batch jobs, monitor progress, move files off XT3 to analysis and viz cluster, archive, steer computation, ... Ex: Fusion simulation, Astrophysics (supernova simulation)
- **“(Real-time) analysis pipelines”**: processing of environmental and earth science data from sensor networks

SDM Tutorial, EDBT'06, Gertz, Ludäscher

Commercial & Open Source Scientific Workflow Systems

Lists Nexus files to process (project)

Reads text files

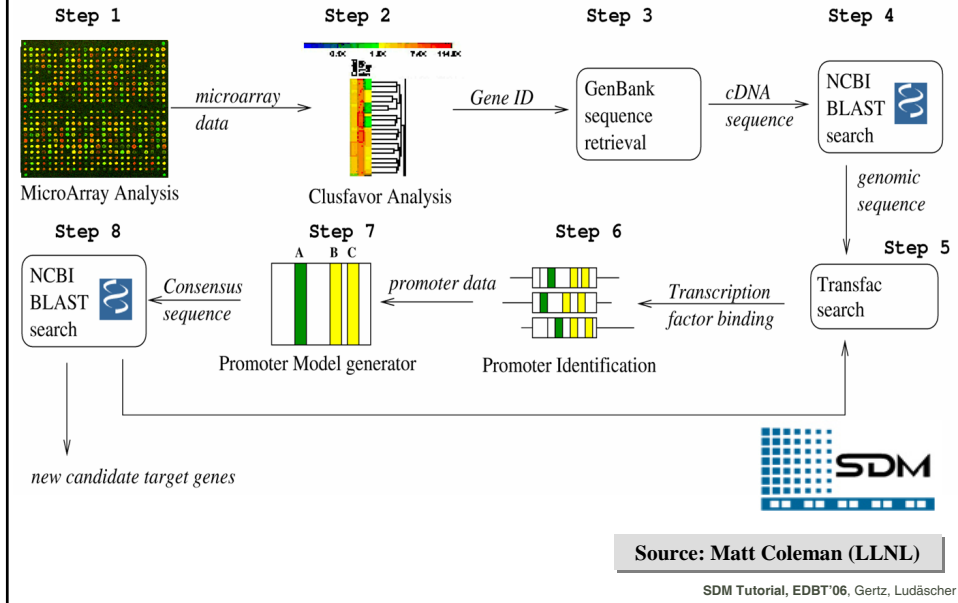
Parses Nexus format

Draws phylogenetic trees

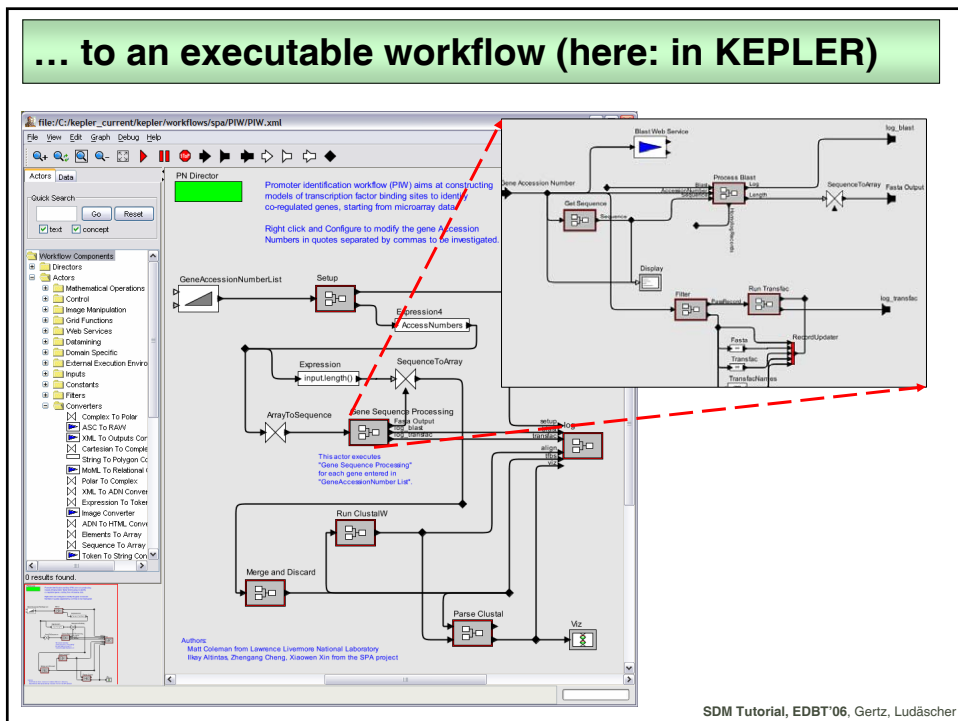
SDM Tutorial, EDBT'06, Gertz, Ludäscher

Promoter Identification Workflow (PIW)

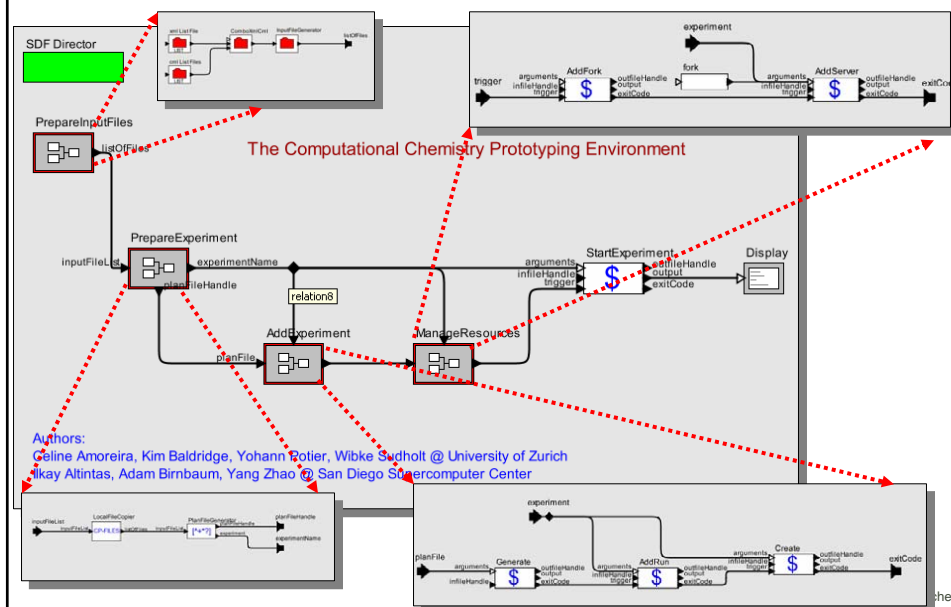
... or from a napkin drawing ...



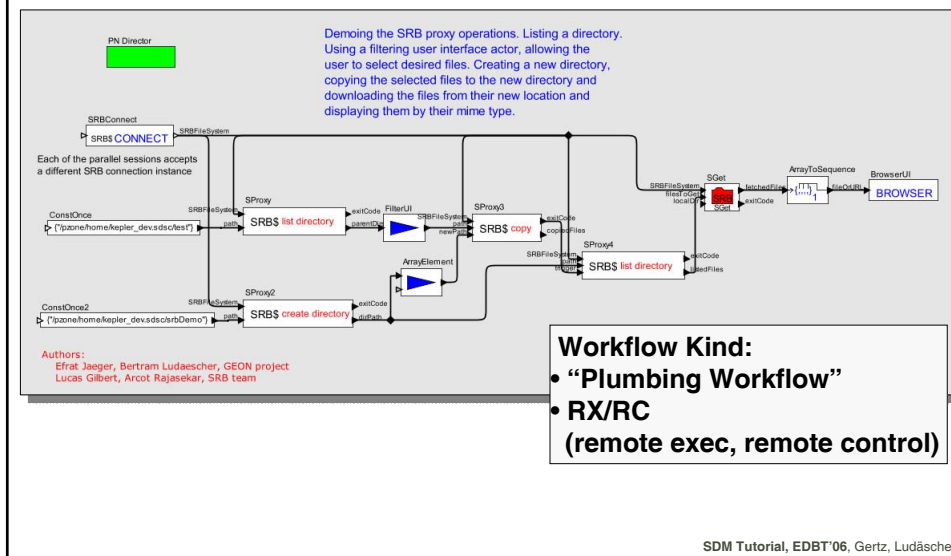
... to an executable workflow (here: in KEPLER)



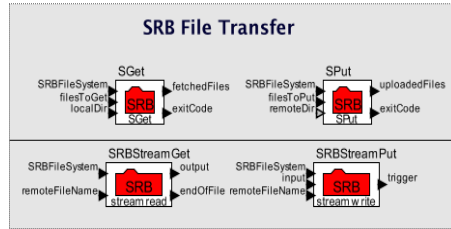
... to a plumbing workflow (Job Mgmt w/ NIMROD)



... more plumbing ...

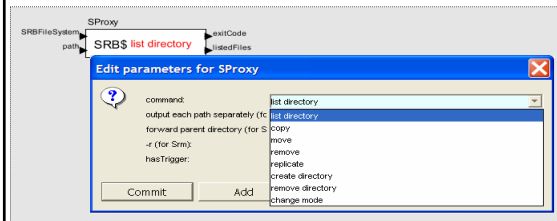


... more plumbing ...



SPut/**S**Get use parallel put/get approaches as provided by JARGON API

Streaming Actors **Stream Put/Get** read & write files from and to SRB as sequence of byte arrays.



SRB Proxy Operations

SDM Tutorial, EDBT'06, Gertz, Ludäscher

The screenshot shows the ROADnet software interface. At the top, it displays "realtime revelle" and "Online realtime data from the Research Vessel Roger Revelle". The main window shows a workflow diagram with the following components:

- Actors:** A list of actors on the left, including `OrbImageSource` and `ImageDisplay`.
- Workflow:** A central diagram showing `OrbImageSource` connected to `ImageDisplay` via an `output` and `input` port. A red dashed arrow points from `OrbImageSource` to an `ORB` database icon.
- Text:** "Streaming demo: images from SIO research vessel accessed via 'bohemia.splorg.org:6580'" and "Tobin T. Fricke, University of California, July 2004".
- Visuals:** A map of the Pacific Ocean showing the vessel's path, a photo of the Research Vessel Roger Revelle, and a video frame showing a whale.
- Browser:** A Mozilla Firefox browser window in the top right showing the vessel's current location and position as of 2-Aug-04.

SDM Tutorial, EDBT'06, Gertz, Ludäscher

Why not just a Python script?

- **Users who can define, reuse, modify, specialize workflows may not be able to do the same (or as easily as) for Python scripts**
- **Other advantages to scientific workflows**
 - Modular reuse and application interoperability
 - Debugging and monitoring workflow execution
 - Automated data management (e.g., provenance)
 - Validation (e.g., data/structural/semantic typing)
- ... From integrated modeling to execution, optimization, archival, etc

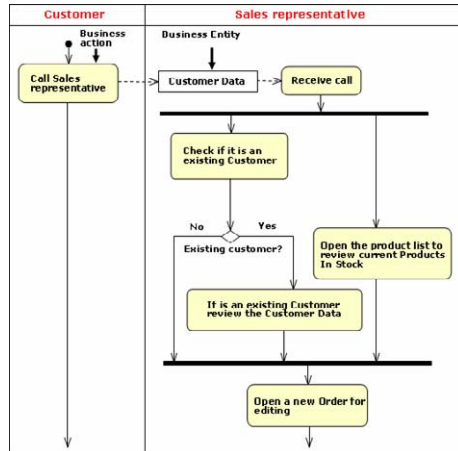
SDM Tutorial, EDBT'06, Gertz, Ludäscher

Business workflows *born-again*?

- **Yes, there are similarities**
 - And we can learn from BWF – e.g. transactions!
- **But also big differences:**
 - **Scientific Workflows:**
 - dataflow oriented
 - streaming/pipelined execution
 - cf. signal processing
 - popular Model of Computation (MoC):
 - Process Networks (PN), Synchronous DataFlow (SDF), Discrete Events (DE), Continuous Time (CT), ...
 - **Business Workflows:**
 - task- and control-flow oriented
 - popular MoC (for theory, abstraction, modeling):
 - Petri-Nets, CSP!?, ...

SDM Tutorial, EDBT'06, Gertz, Ludäscher

Sample Business Workflow

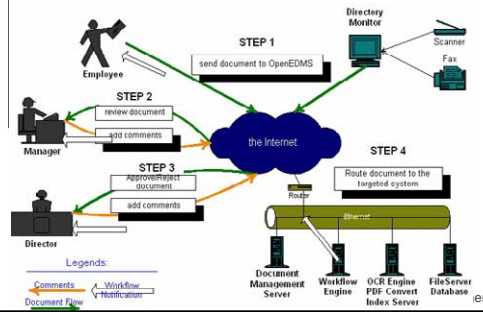


Focus is on ...

- Tasks
- Control-flow
- Work items

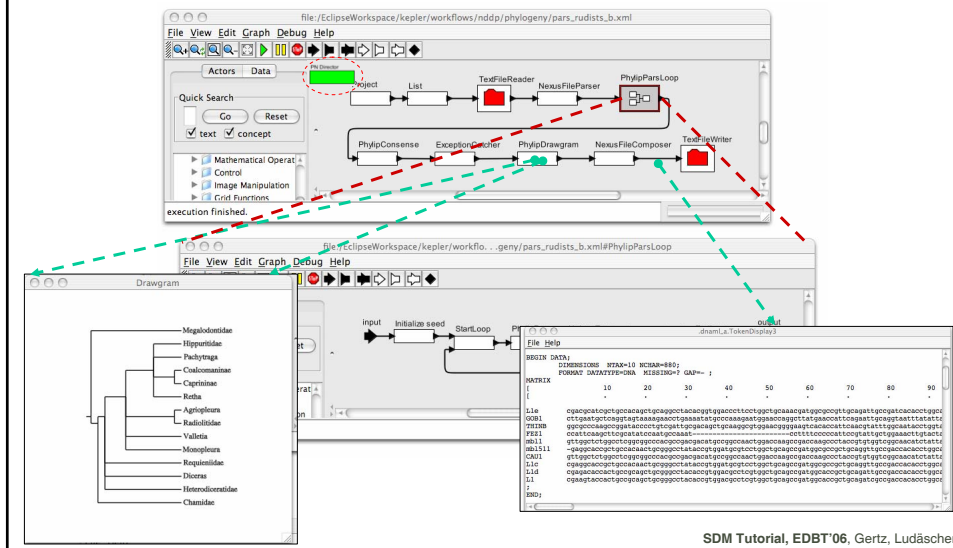
Useful stuff:

- Transactions!
- How to handle complex control-flow ...



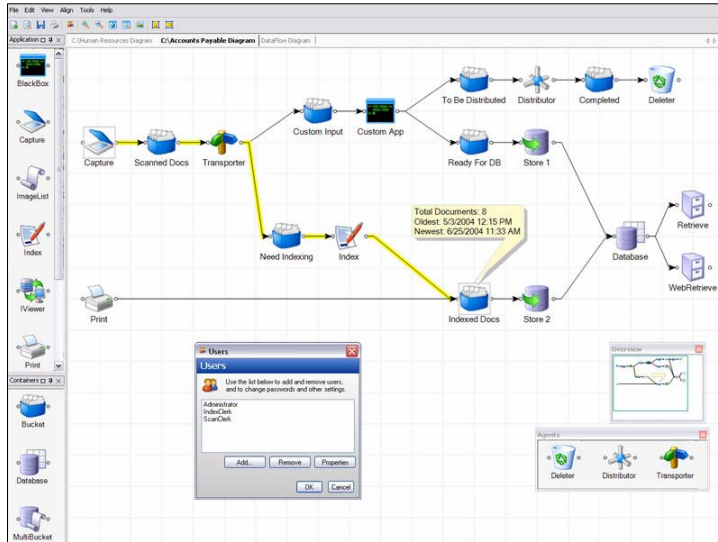
A Simple Scientific Workflow

Example scientific workflow run, executed as a Dataflow Process Network



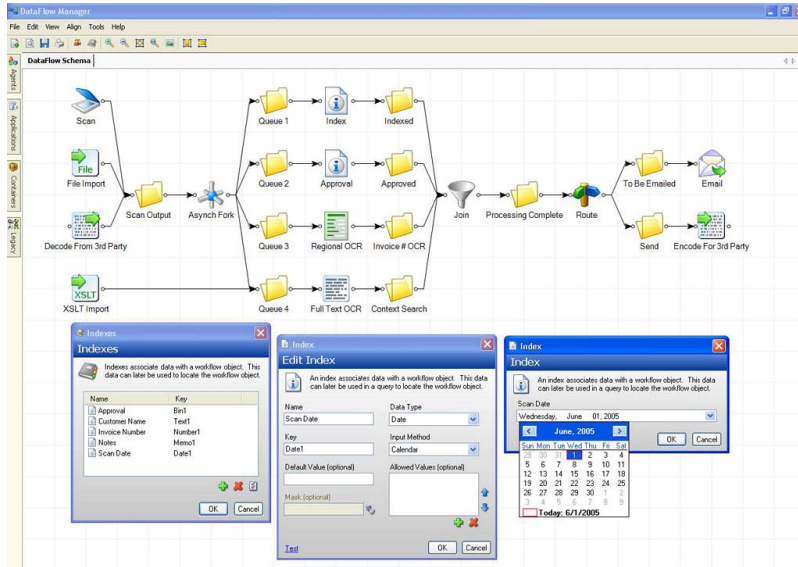
SDM Tutorial, EDBT'06, Gertz, Ludäscher

Pop Quiz! BWF? SWF?



SDM Tutorial, EDBT'06, Gertz, Ludäscher

And the answer is ...



SDM Tutorial, EDBT'06, Gertz, Ludäscher

Wrong Question! (not just SWF vs. BWF, but Dataflow and Data-orientation vs. Control-Flow/Orientation)

DataFlow Manager
Visual Workflow Configuration, Business Rules Definition, Process Monitoring

DataFlow Manager is a strategic data management and workflow product that is easy to use, even with the most complex of workflow systems. The visual, drag-and-drop interface allows system administrators to quickly and easily set up comprehensive workflow process diagrams complete with work rules.

Routing of items is controlled by DataFlow Manager, which allows complex workflows or "schemas" to be visually configured as a combination of independent workflow agents and routing rules. Its companion product, **DataFlow Engine**, provides logging, auditing, production monitoring and comprehensive productivity statistics, including information about the item quantities at various stages in the process.

Server Products

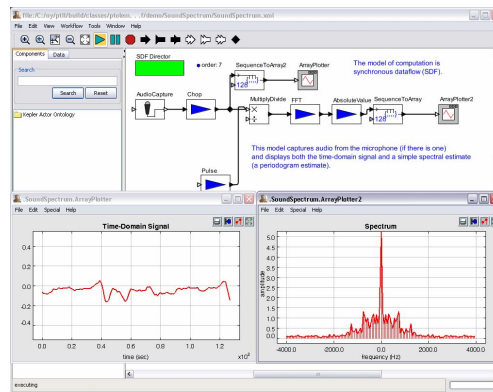
- DataFlow Archive
- DataFlow COLD
- DataFlow Engine
- DataFlow Fax
- DataFlow ImageList
- DataFlow Manager

Visual Workflow Creation | Index Definition | Options | Printing | Notes

SDM Tutorial, EDBT'06, Gertz, Ludäscher

Driving the point home...

- **Dataflow-oriented scientific workflows have features of**
 - ... stream-processing
 - ... data-, task-, and **pipeline-parallelism**
 - ... signal processing systems
 - ... visual PSEs: AVS/Express, IBM DataExplorer, OpenDX, LabView, ...



SDM Tutorial, EDBT'06, Gertz, Ludäscher

Some Features of Scientific Workflows

- **Actor-oriented, Hierarchical Modeling & Design**
 - study interactions of MoCs, compose, nest WFs, ...
 - simulate those systems
- **Automated Provenance System**
 - Keep track of data & workflow provenance
 - “smart” re-run, crash recovery, live & post-mortem analysis & debugging ...
- **Flow-Based Programming Paradigm**
 - data-stream oriented processing
- **Basis for End-to-End Experiment Life-Cycle Management**
 - from design, semantic types, to monitoring & control (“dashboard”), and optimization

SDM Tutorial, EDBT'06, Gertz, Ludäscher

Some Basic SWF Terminology (evolving)

- **Workflow definition W (\approx WF graph we see)**
 - **partial** specification of a workflow (cf. *program*)
 - **parameters P** need to be **instantiated**
 - **data-bindings D** can be viewed as special parameters
- **Model of Computation (MoC – PT terminology)**
 - Looking at W, P, D we still do **not** know how to execute $W(P,D)$ to compute result R
 - A MoC is an algorithm telling us how to apply W on P and D to obtain R .
 - Examples:
 - MoC PN (Process Network):
 - Network of independent processes, communicating through (infinite) unidirectional buffers (queues), prefix-monotonic behavior; given a PN and an input stream and prefix-monotonic, deterministic actors, the output stream is determined! (lots of flexibility for execution!)
 - MoC SDF (Synchronous Dataflow):
 - Similar to PN, but actors must statically declare their token production/consumption rates; solving for pos. int. solutions of balance equations (“LGS”) yields static schedule guaranteeing fixed buffer size

SDM Tutorial, EDBT'06, Gertz, Ludäscher

Some Basic SWF Terminology (evolving)

- **WF Run:** completed computation
- **WF Execution:** ongoing computation
- **Computation graph:** graph data structure keeping track of which token has been computed from which other one(s)
 - Simple examples: evaluating an arithmetic expression; running a “job DAG”
 - But keeping track of “real dependencies” can be tricky
 - Ex: output tuples of an SQL query have “witness tuples” in multiple relations; clear for positive existential queries; what are witnesses for universal and negated queries? $R = A \setminus B$; witnesses anybody?
 - Similar to the notion of “proof tree” in logic (and LP); negation-as-failure looms it’s ugly (beautiful?) head!

SDM Tutorial, EDBT’06, Gertz, Ludäscher

Research Area: Provenance

- **(Abstract) Use Cases:**
 - “Total Recall”: capture everything the MoC can observe
 - ... and more: MoC-inherent plus addtl. **observables**
 - Example: time-stamp *token-in*, *token-out* events → benchmark actor exec time, data movement time, ...
 - The 7 W’s: Who, What, Where, Why, When, Which, (W)how (C. Goble)
 - Smart Re-run:
 - after Pause or Stop, followed by parameter changes: rerun only relevant parts
 - Fault tolerance, crash recovery (cf. checkpointing)
 - Result interpretation and post-mortem analysis

SDM Tutorial, EDBT’06, Gertz, Ludäscher

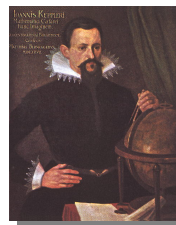
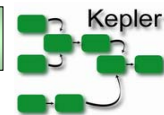
Research Area: Provenance (2)

- **Research Questions:**
 - Given a use case (as a query U) and a provenance schema PS, can U be answered using PS?
(related to *query answering using views* – a reasoning problem!)
 - Ultimately: design PS with U in mind! Also: optimize/specialize PS if U is known/limited
 - Note: the MoC can make a difference! For example, some MoCs have explicit notion of “firing” or might exploit actor declarations (“I’m a function! I have no state!”) This info is relevant e.g. for checkpointing (Need to save state or not? When to save state..)

SDM Tutorial, EDBT'06, Gertz, Ludäscher

An Example System & Project

- **Open-source cross-project collaboration:**
 - NSF/ITR SEEK, GEON, DOE SciDAC/SDM, ...
 - based on Ptolemy II Modeling & Simulation system
 - R&D at Berkeley, SDSC, UCSB, NCSU, UCD, LLNL, Utah, Rutgers, Penn State, Zurich, ...



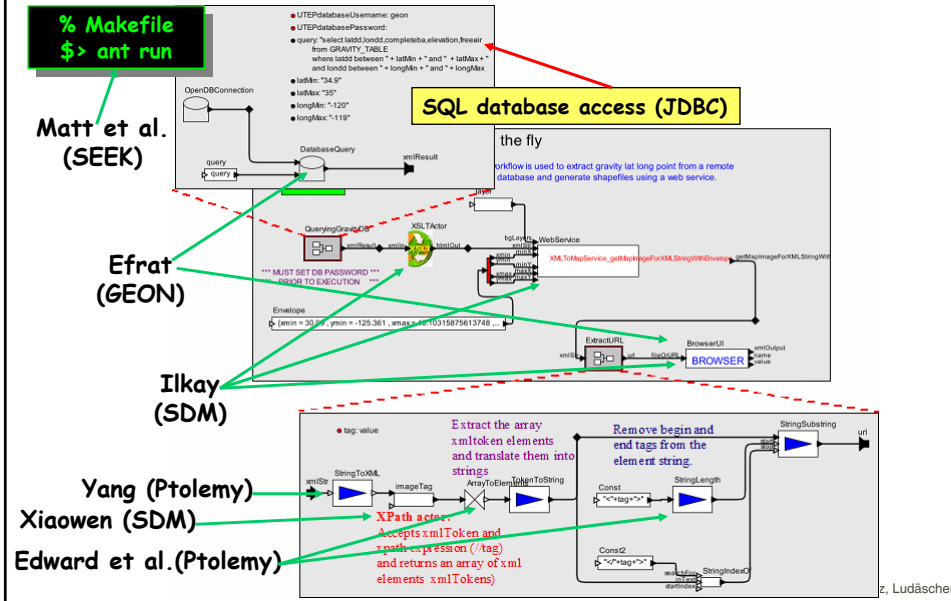
www.kepler-project.org

Collab. tools: IRC, cvs, Wiki, FAQs, ..



SDM Tutorial, EDBT'06, Gertz, Ludäscher

GEON Dataset Generation & Registration (and co-development in KEPLER)



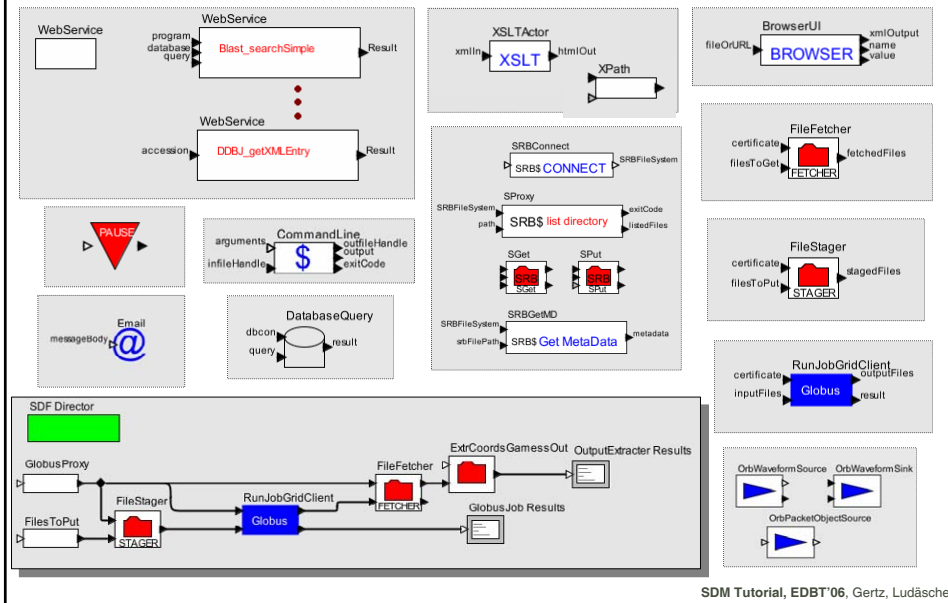
Web Services → Actors (WS Harvester)

Name	URL	Document	Registrant
Blast	http://html.nig.ac.jp/ws/di/Blast.wsdl	document	XML_Central of DDBJ
ClustaW	http://html.nig.ac.jp/ws/di/ClustaW.wsdl	document	XML_Central of DDBJ
DDBJ	http://html.nig.ac.jp/ws/di/DDBJ.wsdl	document	XML_Central of DDBJ
Fasta	http://html.nig.ac.jp/ws/di/Fasta.wsdl	document	XML_Central of DDBJ
TxSearch	http://html.nig.ac.jp/ws/di/TxSearch.wsdl	document	XML_Central of DDBJ

→ "Minute-made" (MM) WS-based application integration

- Similarly: MM workflow design & sharing w/o implemented components

Some KEPLER Actors (out of 160+ ... and counting...)



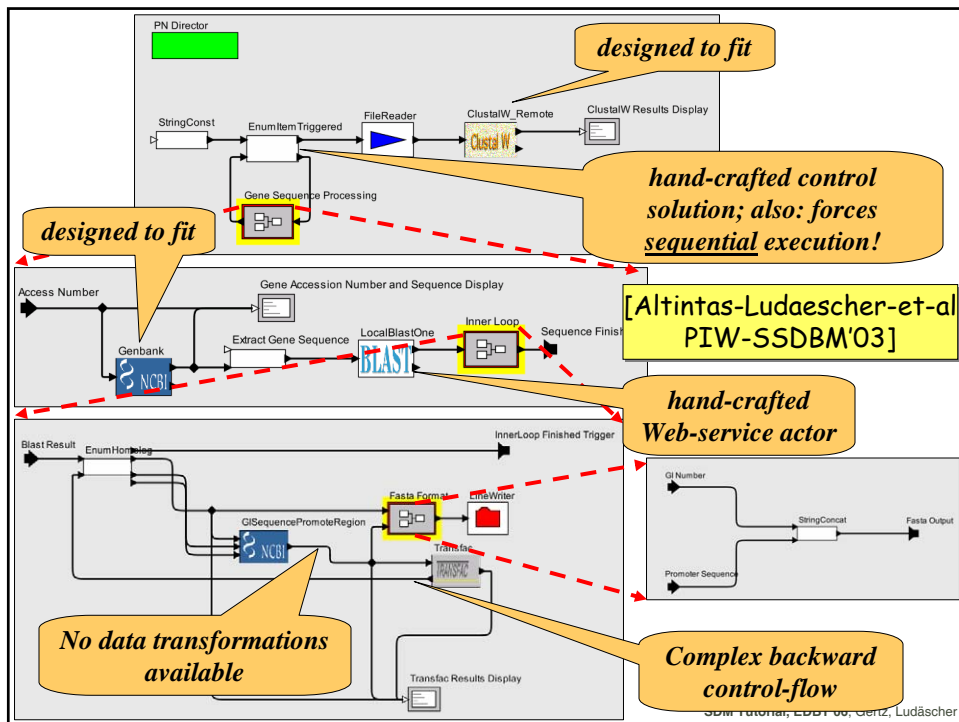
Flow-Based Programming & Design for SWF

- Just doing visual-programming by itself does **not** lead to modular, re-usable, maintainable workflows!
- To fully exploit the dataflow paradigm ...
... **think dataflow!**

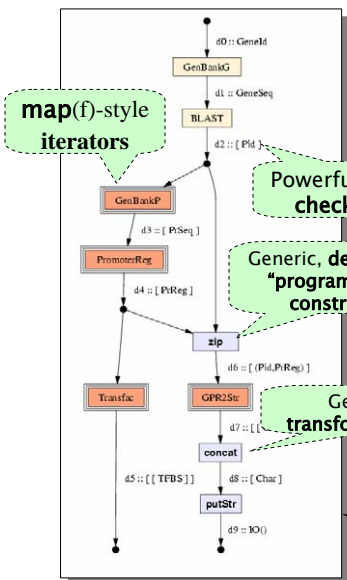
→ Flow-based Programming
→ ... combined w/ Functional,
Collection-oriented Programming



- ... similar to assembly-line metaphor



A Scientific Workflow Problem: More Solved (Computer Scientist's view)



- **Solution based on declarative, functional dataflow process network**
(= also a *data streaming model!*)
- **Higher-order constructs: map(f)**
 - ⇒ no control-flow spaghetti
 - ⇒ data-intensive apps
 - ⇒ free concurrent execution
 - ⇒ free type checking
 - ⇒ automatic support to go from piw(GeneId) to PIW :=map(piw) over [GeneId]

A Scientific Workflow Problem: Even More Solved (domain&CS coming together!)

The screenshot shows the SDF Director interface with a workflow graph. Key components include:

- Const** actor providing input: `{ "NM_001924", "NM_020375" }`
- MAP(GenbankWS)** actor
- MAP(BlankWS)** actor
- MAP(HomologyFilter)** actor
- MAP(Transac)** actor
- WebService** actor: `um.spa.service.Genbank_service`
- ExtractSequence** actor
- GeneSequence** actor
- GeneSequenceArray** actor
- ArrayToSequence** actor
- WebService** actor: `ClustalW_analyzeParam`
- ClustalW Results Display** actor
- Parameters** actor

A callout box shows the output of `map(GenbankWS)`:

```
map(GenbankWS)
Input: {"NM_001924", "NM020375"}
Output: { "CAGT...AATATGAC", "GGGGA...CAAAGA" }
```

Authors: Matthew Coleman from Lawrence Livermore National Laboratory, Ricky Alintas, Bertram Ludäscher, Zhenggang Cheng, Xiaowen Xin from SPA Project, Yang Zhao, Edward Lee from Ptolemy Project.

SDM Tutorial, EDBT'06, Gertz, Ludäscher

Research Problem: Optimization by Rewriting

The graph shows a declarative workflow with the following structure:

- d0 :: GeneId** (input)
- GenBankG** actor
- d1 :: GeneSeq** (output of GenBankG)
- BLAST** actor
- d2 :: [Pid]** (output of BLAST)
- PromoterReg . GenBankP** actor
- d4 :: [PrReg]** (output of PromoterReg)
- zipWith(GPR2Str)** actor
- d7 :: [[Char]]** (output of zipWith)
- Transac** actor
- d5 :: [[TFBS]]** (output of Transac)
- putStr.concat** actor
- d9 :: IO()** (output of putStr.concat)

Annotations:

- map(f o g) instead of map(f) o map(g)** (pointing to the BLAST actor)
- Combination of map and zip** (pointing to the zipWith actor)

- Example: PIW as a declarative, referentially transparent functional process**
 \Rightarrow optimization via functional rewriting possible
 e.g. $\text{map}(f \circ g) = \text{map}(f) \circ \text{map}(g)$
- Technical report & PIW specification in Haskell**

<http://kbis.sdsc.edu/SciDAC-SDM/scidac-tn-map-constructs.pdf>



Optimizing II: Streams & Pipelines

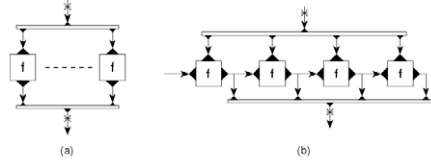


Figure 4.24. Unfolded higher-order functions: a) *map*; b) *scan*

```
(:-) :: α → Streamn α → Streamn α
groupS :: Int → Streamnk α → Streamn (Vectork α)
concatS :: Streamn (Vectork α) → Streamnk α
zipS :: Streamn α → Streamn β → Streamn (α,β)
unzipS :: Streamn (α,β) → (Streamn α, Streamn β)
mapS :: (α → β) → Streamn α → Streamn β
```

Figure 5.10. Types of stream functions

```
zipWithS :: (α → β → γ) → Stream α → Stream β → Stream γ
zipWithS f xs ys = mapS (\(x,y) → f x y) (zipS xs ys)

zipOutS :: (α → (β, γ)) → Stream α → (Stream β, Stream γ)
zipOutS f xs = unzipS (mapS f xs)

zipOutWithS :: (α → β → (γ, δ)) → Stream α → (Stream β, Stream γ, Stream δ)
zipOutWithS f xs ys = unzipS (mapS (\(x,y) → f x y) (zipS xs ys))

iterateS :: (α → α) → α → Stream α
iterateS f a = let ys = a :- (mapS f ys) in xs

generateS :: (α → (α, β)) → α → Stream β
generateS f a = let (zs,ys) = zipOutS f (a :- zs) in ys

scanS :: (α → β → α) → α → Stream β → Stream α
scanS f a xs = let ys = zipWithS f (a :- ys) xs in ys

stateS :: (α → β → (α, γ)) → α → Stream β → Stream γ
stateS f a xs = let (zs,ys) = zipOutWithS f (a :- zs) xs in ys
```

Figure 5.12. Process constructor definitions

- Clean functional semantics facilitates *algebraic workflow (program) transformations* (Bird-Meertens); e.g. $\text{mapS } f \bullet \text{mapS } g \Rightarrow \text{mapS } (f \bullet g)$

SDM Tutorial, EDBT'06, Gertz, Ludäscher

Higher-Order Construct Demo

SDF Director

Const ["5","163"] → IterateOverArray → Display

Given a list Hs of highways, compute a list of results
 $Rc := \text{map}(\text{traffic_info_web}(Hs))$
 I.e., invoke a traffic info web service for each highway in Hs

Author: Ilkay Altintas, Bertram Ludäscher @ SDSC.edu

Inside the 'map' actor: the Traffic-Info web service

Director → port → hwyNums → CATrafficService_getTrafficStatus → port2 → Email

Send a "separate" email for each invocation!

messageBody @

Traffic info for a list of highways: Uses iterate (higher-order "map") actor to access highway info web service repeatedly, sending out one email per highway.

Higher-Order Construct Demo

SDF Director

Const ("5", "163")

IterateOverArray

Display

Given a list Hs of highways, compute a list of results
 Rr = map(traffic_info_we)(Hs)
 I.e., invoke a traffic info web service for each highway in Hs
 Author: Ikkay Alintias, Bertram Ludascher @ SDSC.edu

executing

Inside the 'map' actor: the Traffic-Info web service

Director

port1

port2

messageBody

Send a "separate" email for each invocation!

Email

execution finished

[" reported as of Tuesday, 3
 Slow for the Cone Zone
 I 5
 (SAN DIEGO & IMPERIAL CO
 THE NORTHBOUND & SOUTHBO
 (SAN DIEGO CO) ARE CLOSED FR
 MONDAY THRU FRIDAY THRU 7/16

Traffic info for a list of highways: Uses iterate (higher-order "map") actor to access highway info web service repeatedly, sending out one email per highway.

Higher-Order Construct Demo

SDF Director

Const ("5", "163")

IterateOverArray

Disp

Given a list Hs of highways, compute a list of results
 Rr = map(traffic_info_we)(Hs)
 I.e., invoke a traffic info web service for each highway in Hs
 Author: Ikkay Alintias, Bertram Ludascher @ SDSC.edu

executing

Inside the 'map' actor: the Traffic-Info web service

Director

port1

port2

messageBody

Send a "separate" email for each invocation!

Email

multivac.sdsc.edu - SecureCRT

1 kepler@sdsc.edu Jul 13 45/1831 "Notification email from Keptel"

2 N kepler@sdsc.edu Jul 13 10/327 "Notification email from Keptel"

— VM 6:43: mirrored (INBOX marked) Summary 2 (of 2) new All

From: kepler@sdsc.edu

To: Ludascher@sdsc.edu

Subject: Notification email from Keptel

Date: Tue, 13 Jul 2004 12:12:19 -0700 (PDT)

Content-Type: text/plain; charset=us-ascii

Content-Transfer-Encoding: 7bit

reported as of Tuesday, July 13, 2004 at 12:09 .

Slow for the Cone Zone

SR 163

(SAN DIEGO & IMPERIAL CO'S)
 IS CLOSED FROM THE JCT OF I 5 TO UNIVERSITY AVE /IN SAN DIEGO/
 (SAN DIEGO CO) FROM 2200 HRS EACH NIGHT TO 0500 HRS EACH MORNING MONDAY THRU
 FRIDAY THRU 7/16/04 - DUE TO CONSTRUCTION - A DETOUR IS AVAILABLE

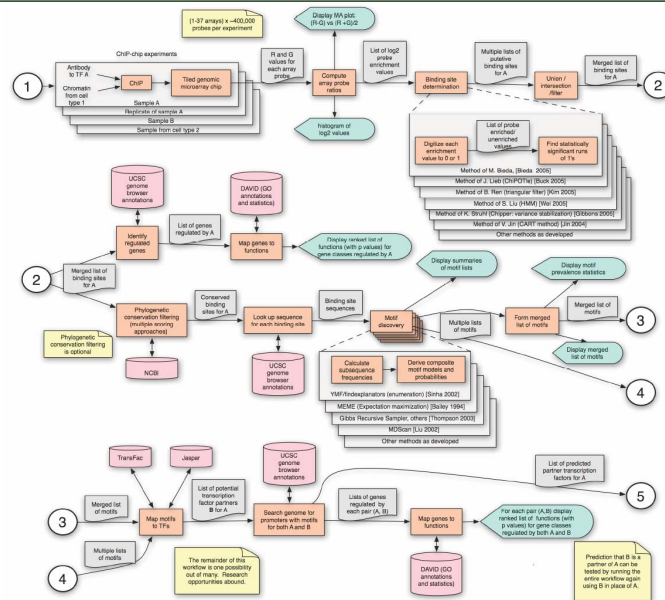
— VM 6:43: mirrored (INBOX marked) 2 (of 2) new All

2 messages, 1 new, 0 unread, 0 deleted

Ready ssh: 3DES 2, 1 30 Rows, 84 Cols VT100

Traffic info for a list of highways: Uses iterate (higher-order "map") actor to access highway info web service repeatedly, sending out one email per highway.

A Bioinformatics Workflow

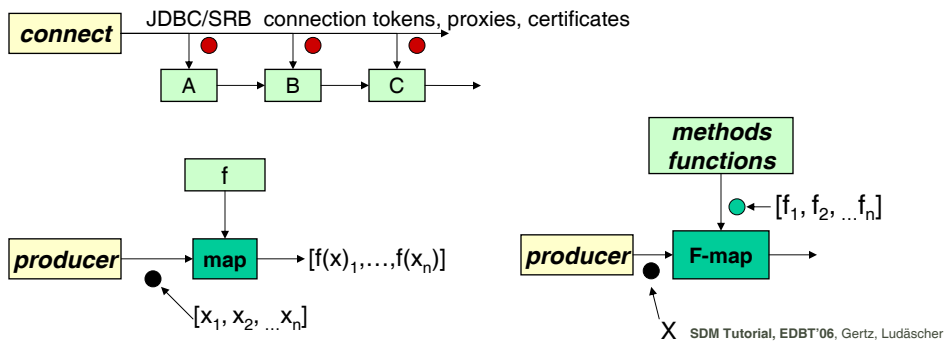


Src: Timothy McPhillips et al.

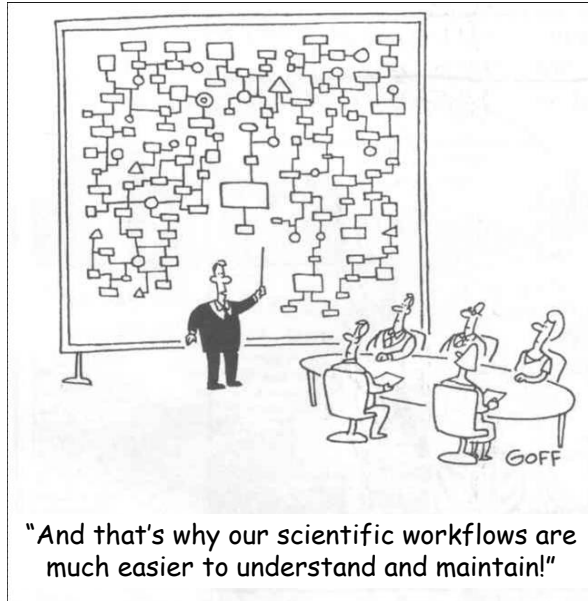
SDM Tutorial, EDBT'06, Gertz, Ludäscher

Towards Flow-based Design Patterns

- **Generality vs specialization of actors**
 - also loosely coupled vs tightly coupled
- **Data transformation pipelines**
 - alternate compute and data transformation steps
- **Stage-execute-fetch pattern (Grid/HPC/HTC-WFs)**
- **Loops, higher-order functions (map, foldr, ...)**
 - cf. Taverna's automatic loop insertion based on data types

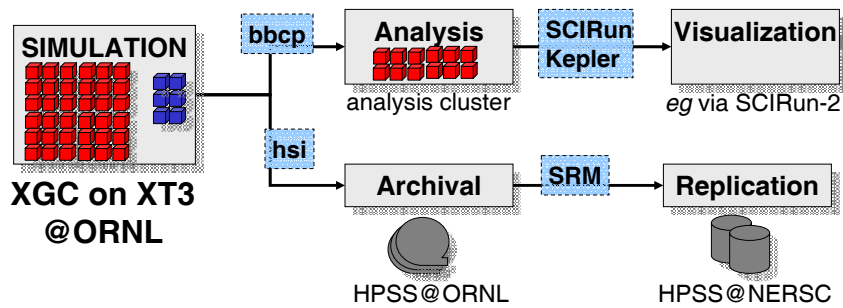


Scientific Workflow Design



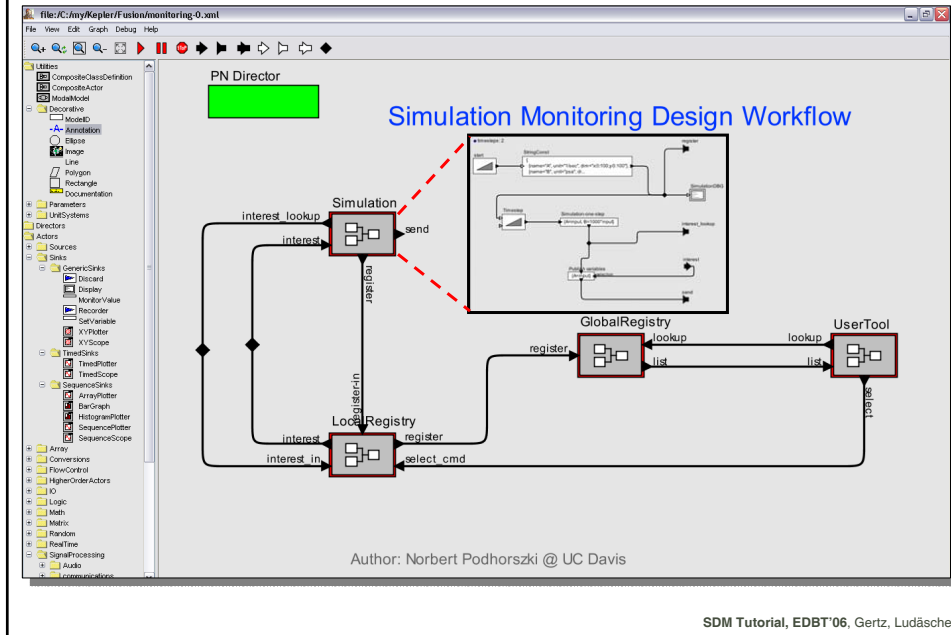
SDM Tutorial, EDBT'06, Gertz, Ludäscher

Simulation Monitoring WF: Design V0

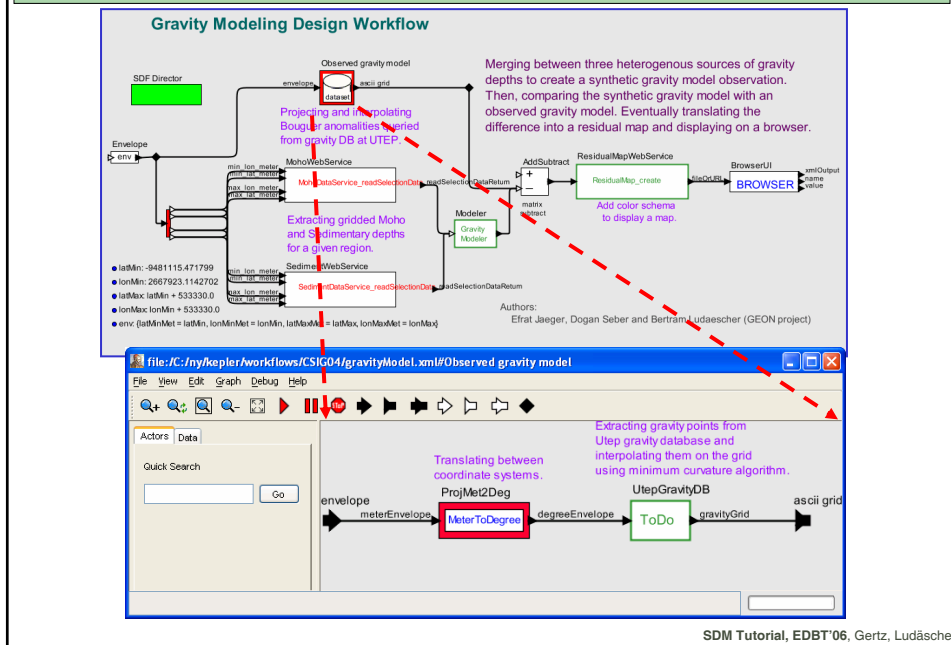


SDM Tutorial, EDBT'06, Gertz, Ludäscher

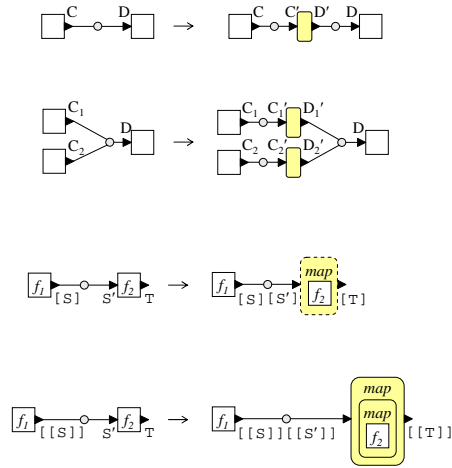
WF Design, Simulation, Prototyping ...



Blurring Design (ToDo) and Execution



WF-Design: Adapters for Semantic & Structural “Gaps”



Adapters may:

- be abstract (no impl.)
- be concrete
- bridge a semantic gap
- fix a structural mismatch
- be generated automatically (e.g., Taverna’s “list mismatch”)
- be reused components (based on signatures)

Source: [Bowers-Ludaescher, ER’05]

SDM Tutorial, EDBT’06, Gertz, Ludäscher

Additional Design Primitives for Semantic Types

Extended Transformations	Starting Workflow	Resulting Workflow	Resulting Workflow
t_9 : Actor Semantic Type Refinement ($T' \sqsubseteq T$)			
t_{10} : Port Semantic Type Refinement ($C' \sqsubseteq C, D' \sqsubseteq D$)			
t_{11} : Annotation Constraint Refinement ($\alpha' \rightarrow \alpha$)			
t_{12} : I/O Constraint Strengthening ($\psi \rightarrow \varphi$)			
t_{13} : Data Connection Refinement			
t_{14} : Adapter Insertion			
t_{15} : Actor Replacement			
t_{16} : Workflow Combination (Map)			

Source: [Bowers-Ludaescher, ER’05]

Scientific Workflow Design

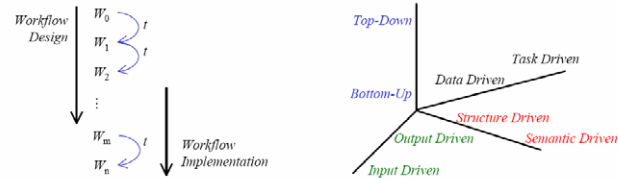


Fig. 2. Workflow engineers evolve workflows by applying design primitives (left), shown as transformations t ; and certain primitives can be grouped to form design strategies (right), where each design strategy is shown as a distinct dimension of a design space.

- **Support SWF design & reuse, via:**
 - Structural data types
 - Semantic types
 - Associations (=constraints) between them
 - Type checking, inference, propagation
- ➔ Separation of concerns:
 - structure, semantics, WF orchestration, etc.

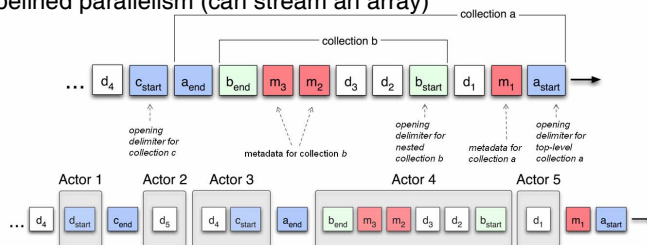
Basic Transformations	Starting Workflow	Resulting Workflow	Resulting Workflow
t_1 : Entity Introduction (actor or data connection)			
t_2 : Port Introduction			
t_3 : Datatype Refinement ($a' \leq a, t' \leq t$)			
t_4 : Hierarchical Abstraction			
t_5 : Hierarchical Refinement			
t_6 : Data Connection			
t_7 : Director Introduction			

Source: [Bowers-Ludaescher, ER'05]

Tutorial, EDBT'06, Gertz, Ludäscher

Collection-Oriented SWF Modeling & Design

- **Assembly line metaphor + Signal Processing + XML + ...**
 - Streams are nested collections (\approx XML)
 - Stream data schema is "registered" to a WF data model (really need this)
 - use ideas from "punctuated streams" to delineate collections
- **Advantages:**
 - Less "messy" WFs (more linear, less branching)
 - "Add-only" mode (inject new derived information); *augmentation instead of transformation*
 - Tagging data for downstream processing (instead of "bombing", pass on "dirty" / faulty / strange data with a relevant tag)
 - Pipelined parallelism (can stream an array)



SDM Tutorial, EDBT'06, Gertz, Ludäscher

Fault Tolerance & Maintenance Challenges

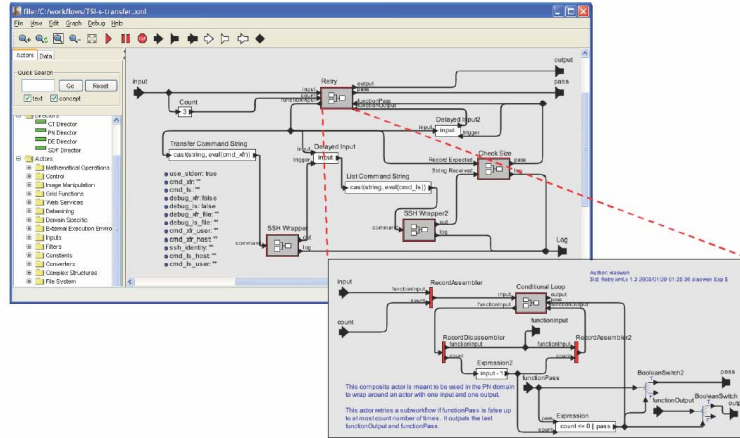


Figure 1: Control-flow intensive astrophysics workflow in KEPLER [21]. “Retry”, a composite actor for fault-tolerant data transfer (top), contains a subworkflow (bottom), which itself contains a “ConditionalLoop” subworkflow (inside not shown). Complex feedback loops and the use of boolean switches illustrate the complexity of modeling control-flow directly in a dataflow process network.

SDM Tutorial, EDBT’06, Gertz, Ludäscher

Workflow Templates and Patterns

New Ingredients

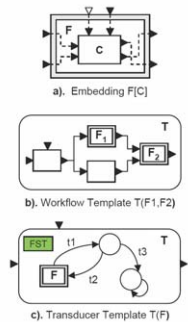


Figure 2: a) Embedding of component C in frame F; b) workflow template T(F₁, F₂); c) finite state transducer template T(F).

work w/ Anne Ngu, Shawn Bowers, Terence Critchlow

Proposed Layered Architecture

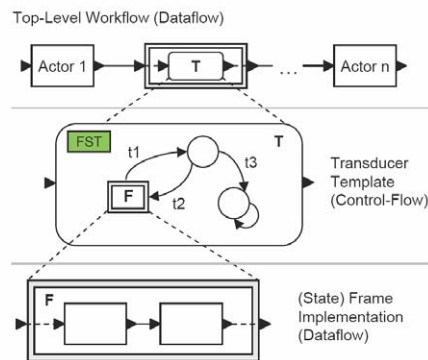


Figure 3: A pattern for modeling generic control-flow components that consists of an outer frame (top), a nested transducer template (middle), and state-frame embeddings (bottom)

SDM Tutorial, EDBT’06, Gertz, Ludäscher

Use of Semantics in SWF...

“Smart” Search

- Concept-based, e.g., “find all datasets containing biomass measurements”

Improved Linking, Merging, Integration

- Establishing links between data *through* semantic annotations & ontologies
- Combining heterogeneous sources based on annotations
- Concatenate, Union (merge), Join, etc.

Transforming

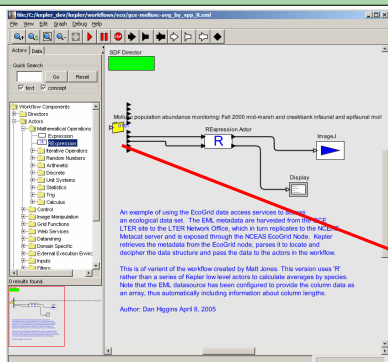
- Construct mappings from schema S1 to S2 based on annotations

Semantic Propagation

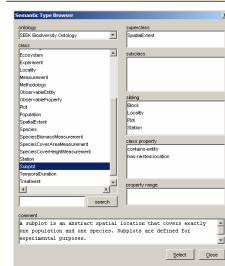
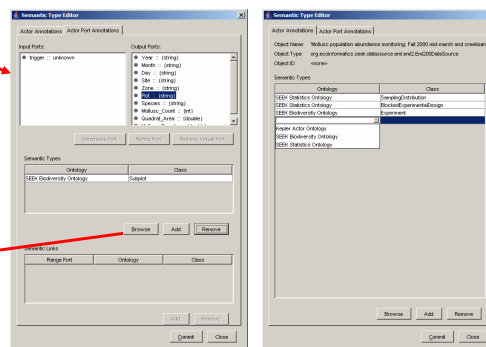
- “Pushing” semantic annotations through transformations/queries

SDM Tutorial, EDBT'06, Gertz, Ludäscher

Typing Workflow Components



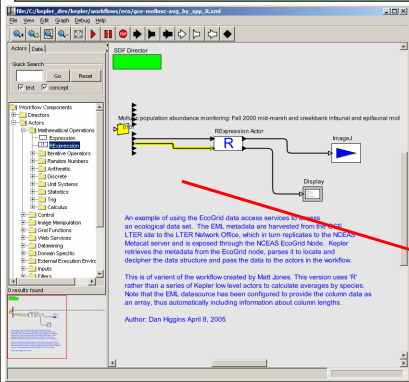
Semantic Type Editor is used to assign one or more semantic types to the component or to the component's input and output ports. In the simplest case, a semantic type is a class taken from an OWL-DL ontology. Multiple types define a conjoined concept expression.



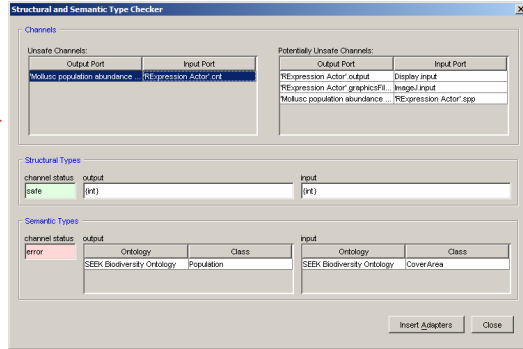
A simple **ontology browser** is provided in Kepler to navigate a classified OWL-DL ontology. Classes can be searched for and selected as a semantic type.

SDM Tutorial, EDBT'06, Gertz, Ludäscher

Checking Type Constraints



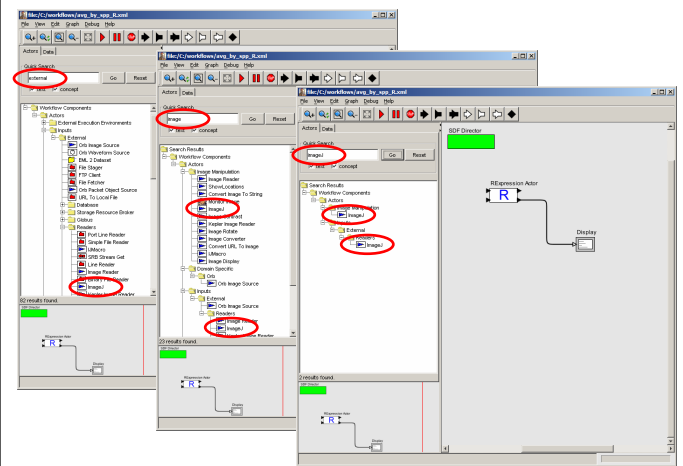
Kepler can **statically perform semantic and structural type checking** of connections. A type checker allows the user to see potentially mismatched port connections as well as known type conflicts before workflow execution.



The user can **navigate the unsafe and potentially unsafe channels** using the Kepler Type Checker dialog. When a channel is selected: (a) it is highlighted on the canvas, (b) the structural type and status is shown (here, the channel is structurally well typed), and (c) the semantic type and status is shown (here, the connection produce a semantic type error).

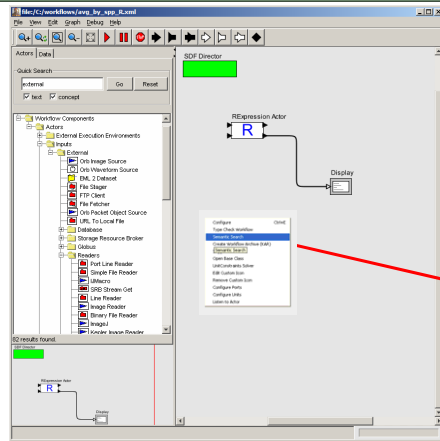
Kepler Actor-Library

- **Ontology-based actor organization / browsing**
- **Customizable libraries based on ontologies**
- **Text search with concept-based expansion**



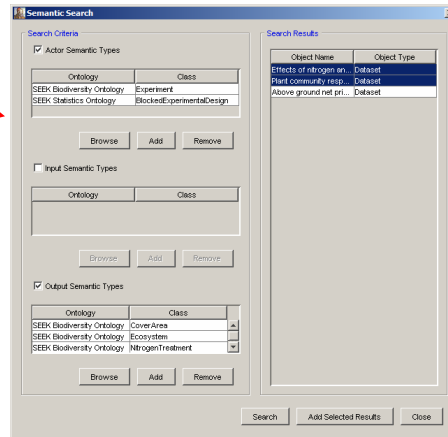
Users can discover ImageJ using various search terms. Here, ImageJ shows up in multiple tree locations based on its given annotations. The library search permits text-based matching against the component's metadata (its given name and certain properties), expanded with concept matches.

Semantic Searching



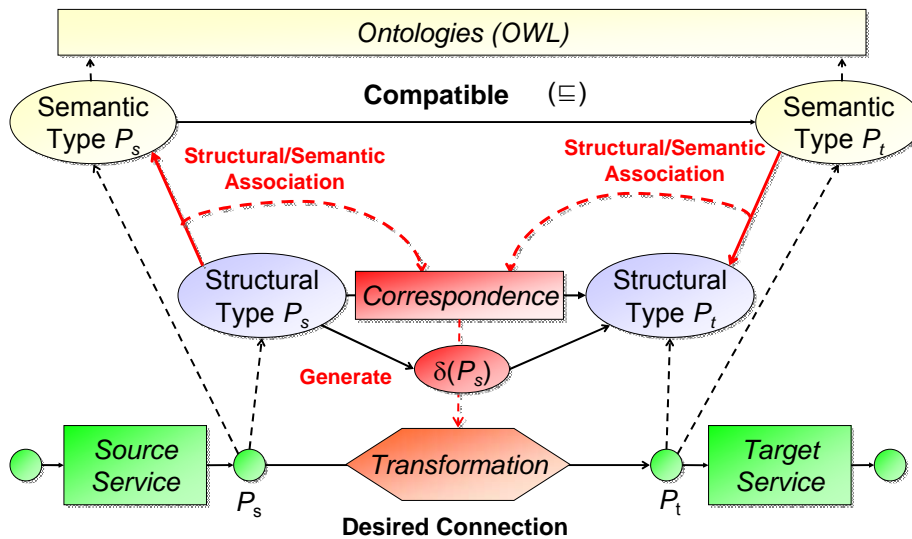
Kepler provides a more advanced ontology-based search mechanism. Users can start the **Semantic Search** dialog, where components can be searched for based on their semantic types.

The Semantic Search dialog allows a user to search components by any combination of actor, input, and output semantic types.



SDM Tutorial, EDBT'06, Gertz, Ludäscher

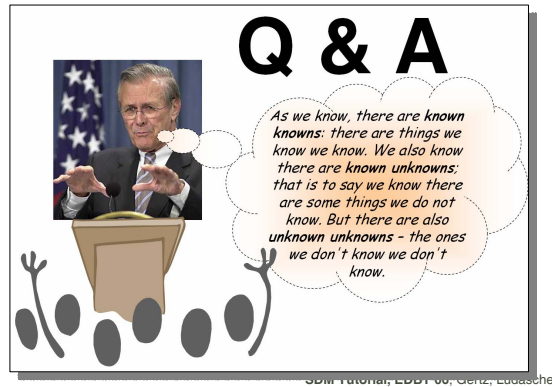
Ontology-Guided Data Transformation



Source: [Bowers-Ludaescher, DILS'04]

Summary: Scientific Workflows

- Scientific Workflows in e-Science and CI
- SWF vs Business Workflows
- Features of a SWF System (Kepler)
- Flow-based Programming and Scientific Workflow Design
- Semantic Extensions

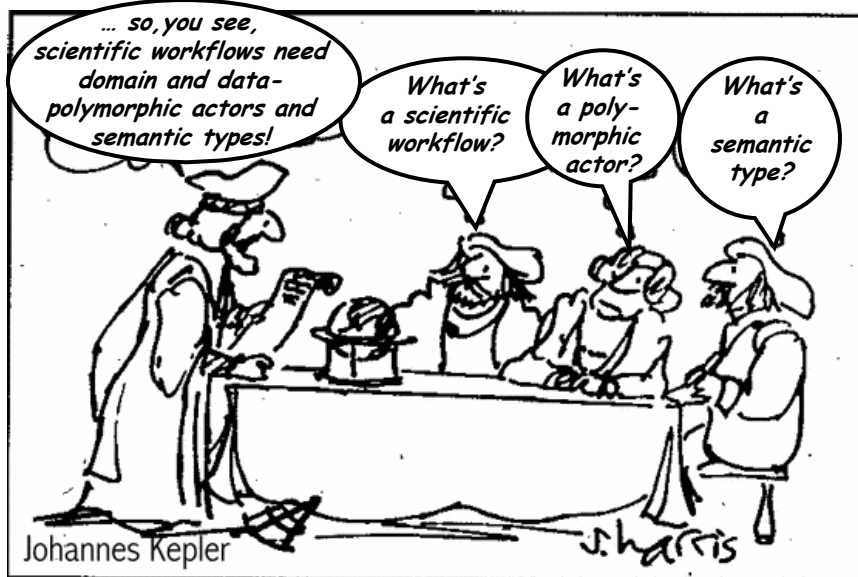


Back then ... KEPLER... was ahead of his time ...



SDM Tutorial, EDBT'06, Gertz, Ludäscher

... but such is life ... ;-)

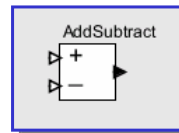


SDM Tutorial, EDBT'06, Gertz, Ludäscher

Polymorphic Actors: Components Working Across Data Types and Domains

- **Actor Data Polymorphism:**
 - Add **numbers** (int, float, double, Complex)
 - Add **strings** (concatenation)
 - Add **complex types** (arrays, records, matrices)
 - Add **user-defined types**
- **Actor Behavioral Polymorphism:**
 - In **dataflow**, add when all connected inputs have data
 - In a **time-triggered model**, add when the clock ticks
 - In **discrete-event**, add when any connected input has data, and add in zero time
 - In **process networks**, execute an infinite loop in a thread that blocks when reading empty inputs
 - In **CSP**, execute an infinite loop that performs rendezvous on input or output
 - In **push/pull**, ports are push or pull (declared or inferred) and behave accordingly
 - In **real-time CORBA***, priorities are associated with ports and a dispatcher determines when to add

*hey, Ptolemy has been out for long!



By not choosing among these when defining the component, we get a huge increment in component re-usability. But how do we ensure that the component will work in all these circumstances?

Source: Edward Lee et al. <http://ptolemy.eecs.berkeley.edu/>