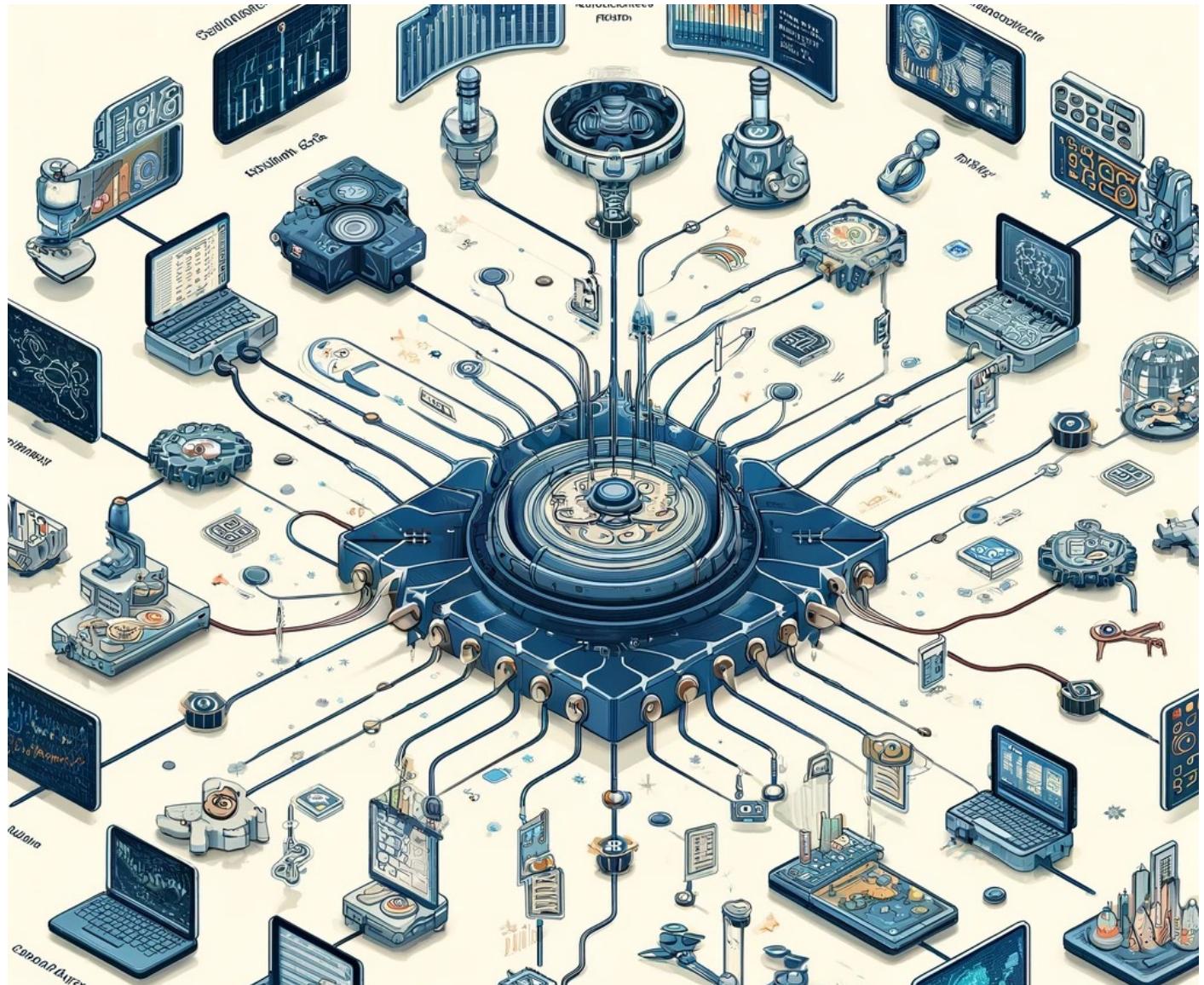


# Exploring Scientific Workflows with CWL and dispel4py

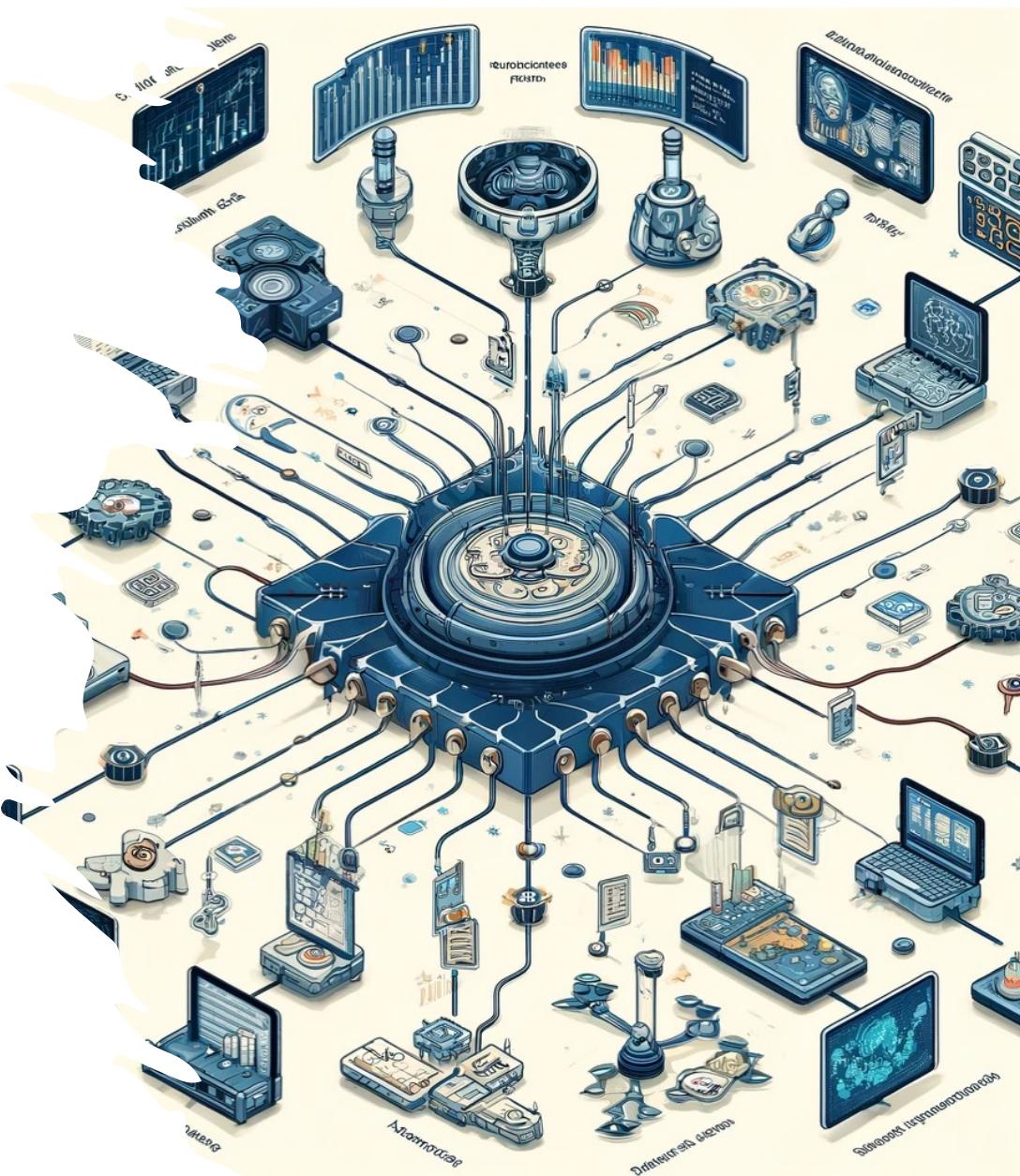
## Module 1.a

- Dr. Rosa Filgueira
- Lecturer at the School of Computer Science
- University of St Andrews
- [rf208@st-andrews.ac.uk](mailto:rf208@st-andrews.ac.uk)
- [rosa.filgueira.vicente@gmail.com](mailto:rosa.filgueira.vicente@gmail.com)



# Seminar Overview

- Day 1: Understanding Scientific Workflows (4 Hours)
    - Module 1: Introduction to Scientific Workflows
      - Module 1.a
        - Defining scientific workflows and WMS
        - How to Select WMS
      - Module 1.b
        - Beyond the Basics Concepts
        - Research Directions
    - Module 2: Creating Workflows with CWL
  - Day 2: Exploring dispel4py and its applications (4 Hours)
    - Module 3: Introduction to dispel4py
    - Module 4: Hands-on Exercises & latest research works



# Module 1.a- Introduction to Scientific Workflows

1. Defining scientific workflows and Workflows Management Systems (WMS)
  - i. Workflow Definition and Importance
  - ii. Historical Context
  - iii. You might use workflows already
  - iv. Workflow Key Phases
  - v. Workflow High-Level Elements
  - vi. Workflow Management Systems (WMS)
2. How to Select WMS
  - i. Key Factors to Consider
  - ii. Interface
  - iii. Workload
  - iv. Community
  - v. Task-Flow vs Data-Flow
  - vi. Real Examples



## 1. Defining scientific workflows & Workflow Management Systems (WMS)

# Workflow Definition and Importance

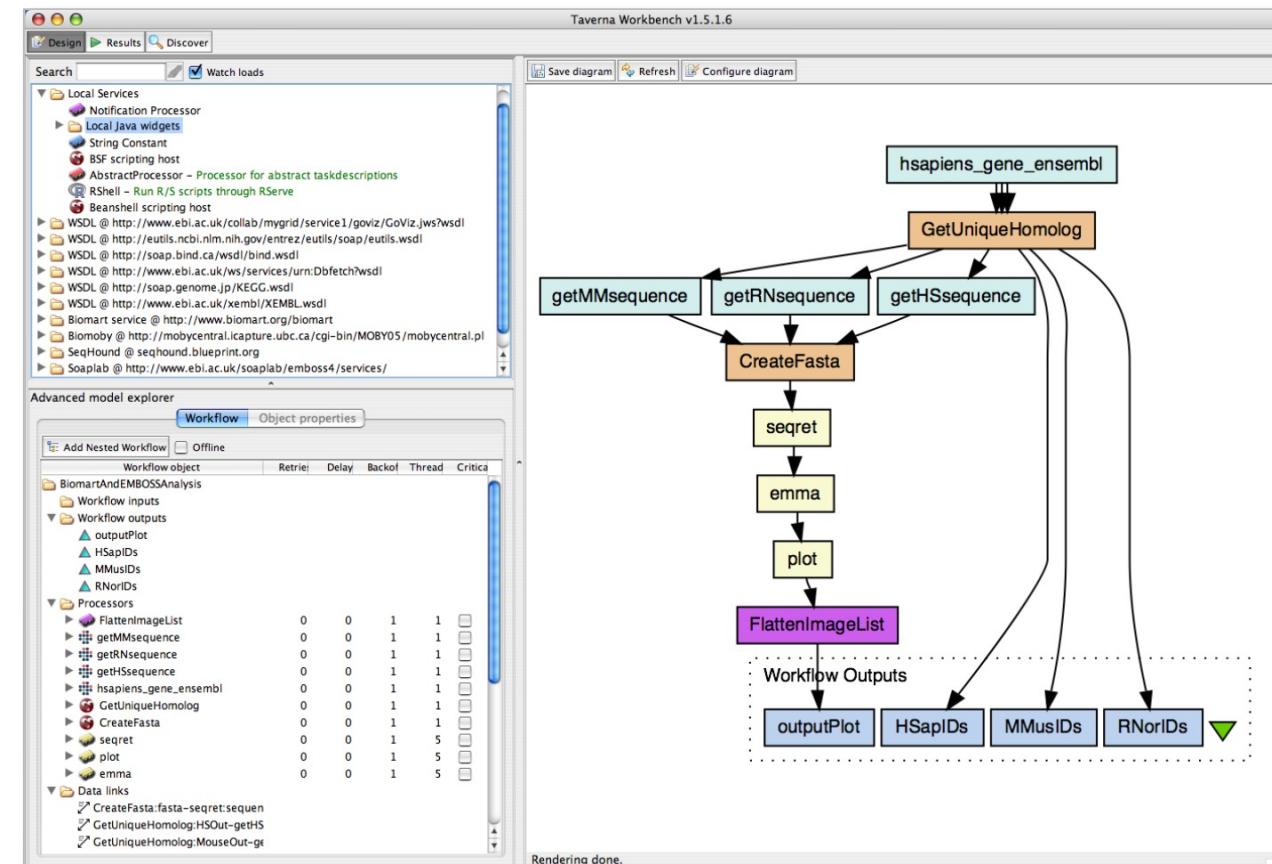
- Workflows refer to the structured sequences of:
  - Data processing
  - Analysis tasks
- Workflows are crucial for ensuring:
  - Systematically execution of complex and multi-step processes/tasks
    - Facilitate the organization of tasks
  - Verifiable and reproducible results.
    - Enhance the efficiency and reliability of scientific research.

# Workflow Definition and Importance

- Formal way to express a calculation
- Multiple tasks with dependencies between them
- No limitations on tasks
  - Short or long
  - Loosely or tightly coupled
- Capture task parameters, input, output
- Independence of workflow process and data
- Often, run same workflow with different data

# Workflow Definition and Importance

- Example workflow represented in the Taverna workflow system.
  - Extracts gene IDs from human chromosome 22 with mappings to disease functions and homologues in mouse and rat
  - Fetches base pairs of the associated DNA sequences;
  - Combines the sequences into a FASTA file;
  - Performs a multiple sequence alignment;
    - soaplab-based operations (seqret, emma, plot)
  - Renders the results



# Historical Context

- The concept of workflows in science has evolved
  - from manual paper-based processes → highly automated systems
- This evolution has been driven by
  - increasing complexity of scientific research
  - processing & analysing large volumes of data.
- Scientific workflows are integral to a wide range of disciplines:
  - BioInformatics
  - Astronomy
  - NeuroInformatics
  - GeoSciences
  - Social Sciences ....

# Historical Context

- What do you do when to want to run computations on a cluster ?
  - you write a submission script and submit it to the scheduler (e.g. Slurm) ?

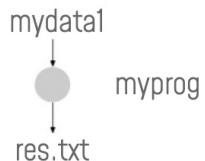
```
#!/bin/bash
# Submission script for demonstrating
# slurm usage.

# Job parameters
#SBATCH --job-name=demo
#SBATCH --output=res.txt
# Needed resources
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu=2000
#SBATCH --time=1:00:00

# Operations
echo "Job start at $(date)"
# Job steps
srun ~/bin/myprog < mydata1

echo "Job end at $(date)"
```

19,0-1      All



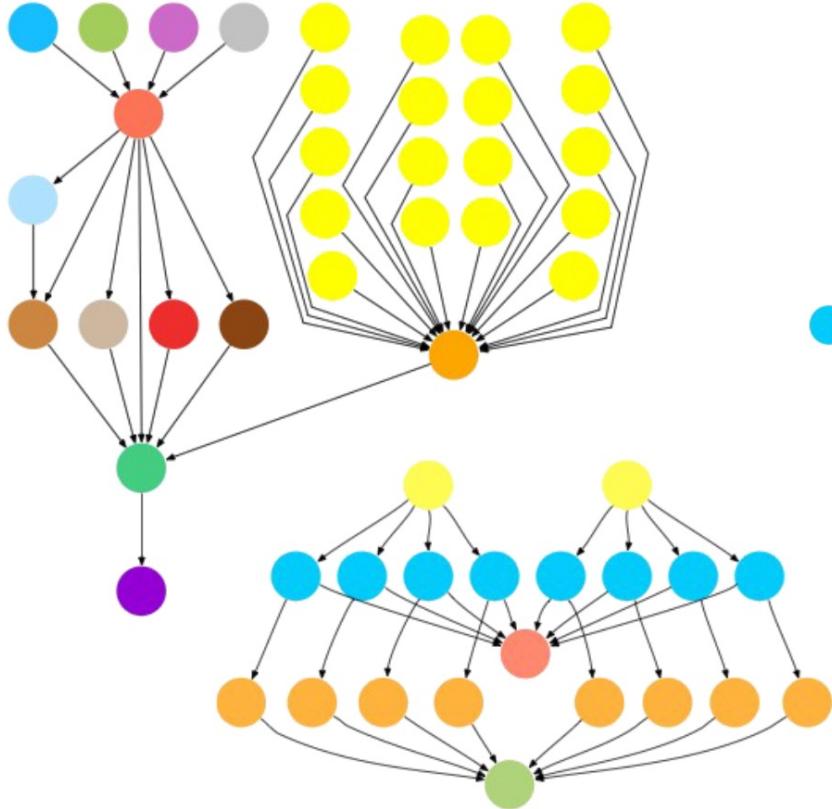
```
#!/bin/bash
# #!/bin/bash
# # #!/bin/bash
# # Submission script for demonstrating
# # slurm usage.

#S #
#S #S # Job parameters
#S #S #SBATCH --job-name=demo
#S #S #SBATCH --output=res.txt
#S #S # Needed resources
#S #S #SBATCH --ntasks=1
#S #S #SBATCH --mem-per-cpu=2000
#S #S #SBATCH --time=1:00:00
# ec#
# ec# Operations
sr# echo "Job start at $(date)"
sr# Job steps
ec srun ~/bin/myprog < mydata1
ec
~ echo "Job end at $(date)"
```

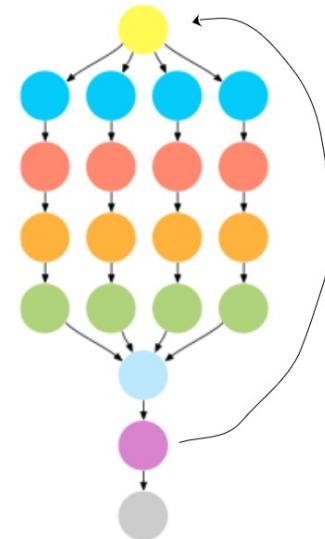
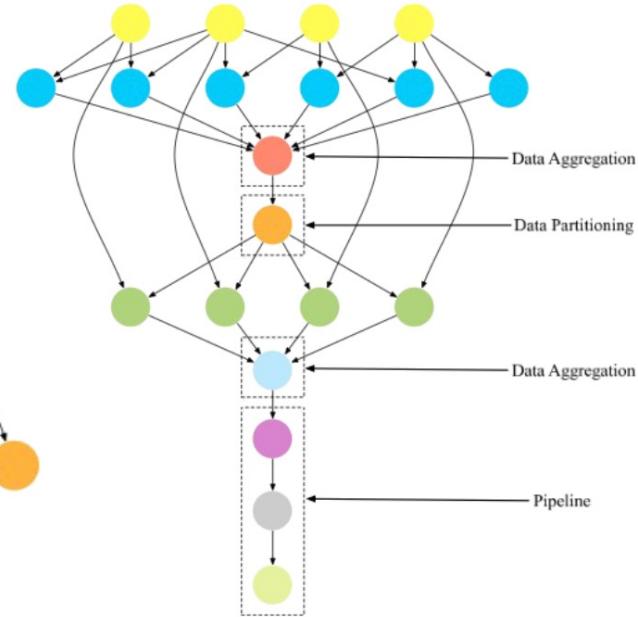
19,0-1      All



# Historical Context



Q: What if ...



You need workflows !

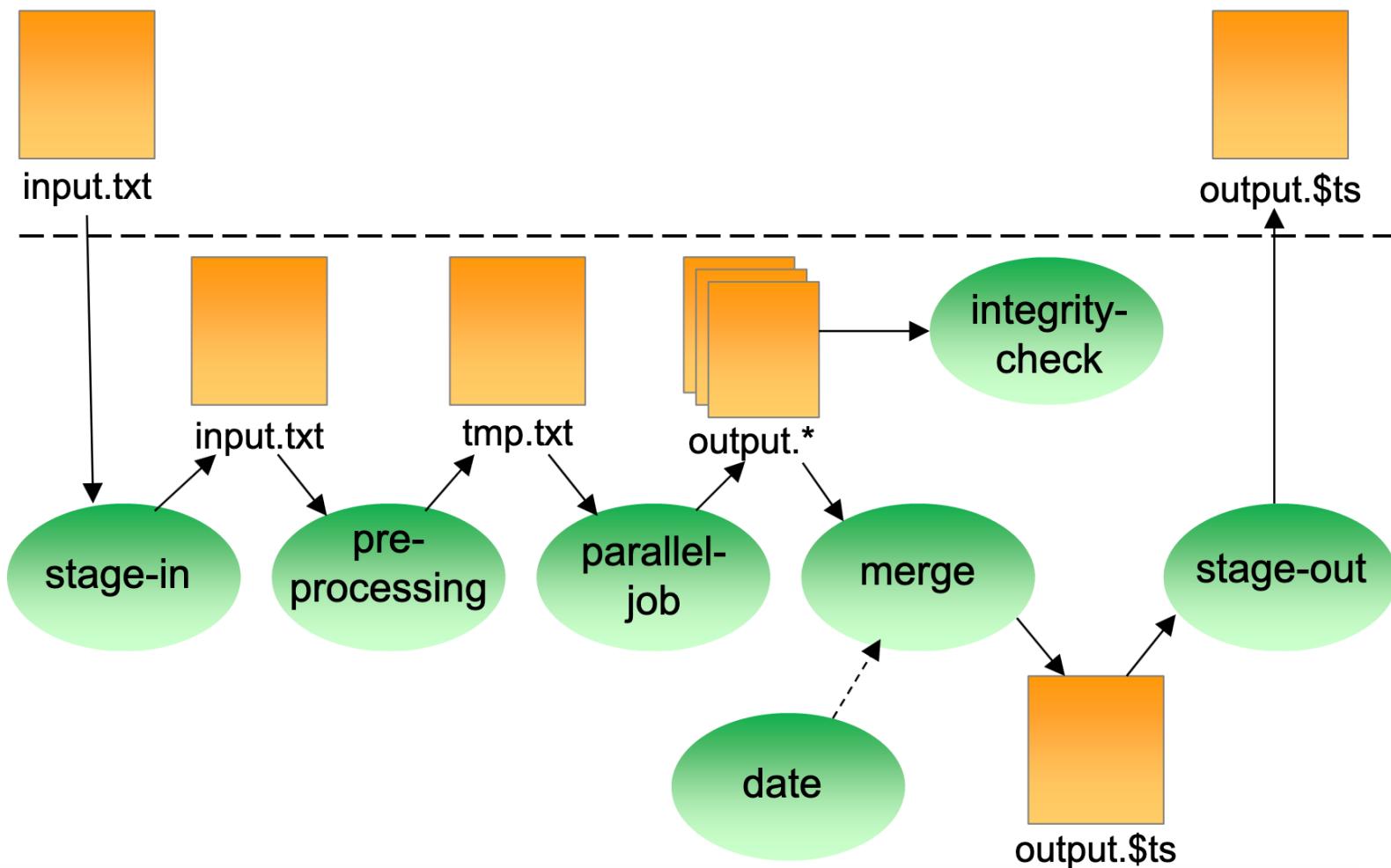
# You might use workflows already

Example of a Shell Script “*Workflow*”

```
#!/bin/bash
1) Stage-in input data to compute environment
scp myself@datastore.com:/data/input.txt /scratch/input.txt
2) Run a serial job with an input and output
bin/pre-processing in=input.txt out=tmp.txt
3) Run a parallel job with the resulting data
mpiexec bin/parallel-job in=tmp.txt out_prefix=output
4) Run a set of independent serial jobs in parallel – scheduling by hand
for i in `seq 0 $np`; do
    bin/integrity-check output.$i &
done
5) While those are running, get metadata and run another serial job
ts=`date +%s`
bin/merge prefix=output out=output.$ts
6) Finally, stage results back to permanent storage
scp /scratch/output.$ts myself@datastore.com:/data/output.$ts
```

# You might use workflows already

Schematic of the Shell Script “Workflow”



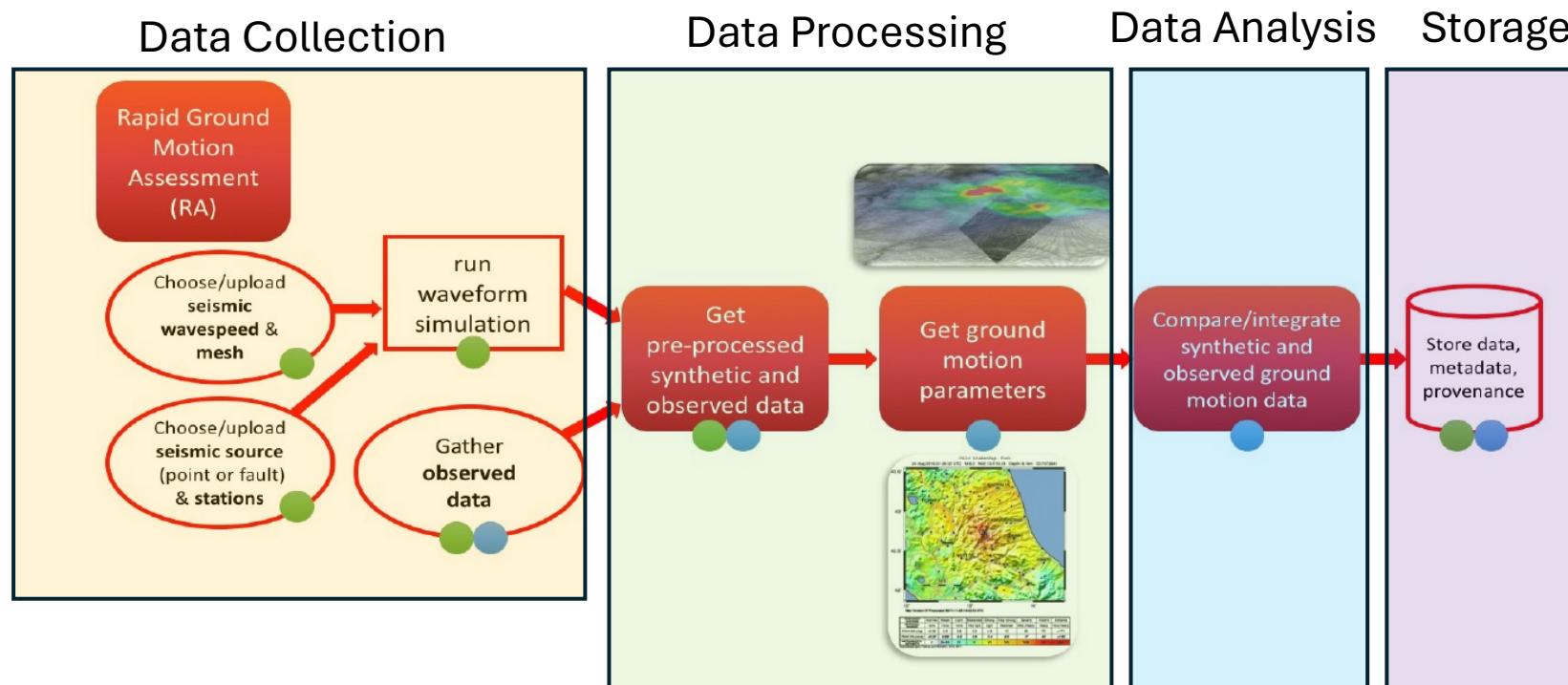
# Workflows Key Phases

- Data Collection
  - acquiring raw data from experiments or simulations
- Data Processing
  - cleaning and structuring data for analysis
- Data Analysis
  - applying statistical models and algorithms to interpret data
- Visualization & Storage
  - creating graphical representations of analysis results
  - storing the results/metadata/provenance for further analysis

These components are interconnected in a workflow. Lets see an example!

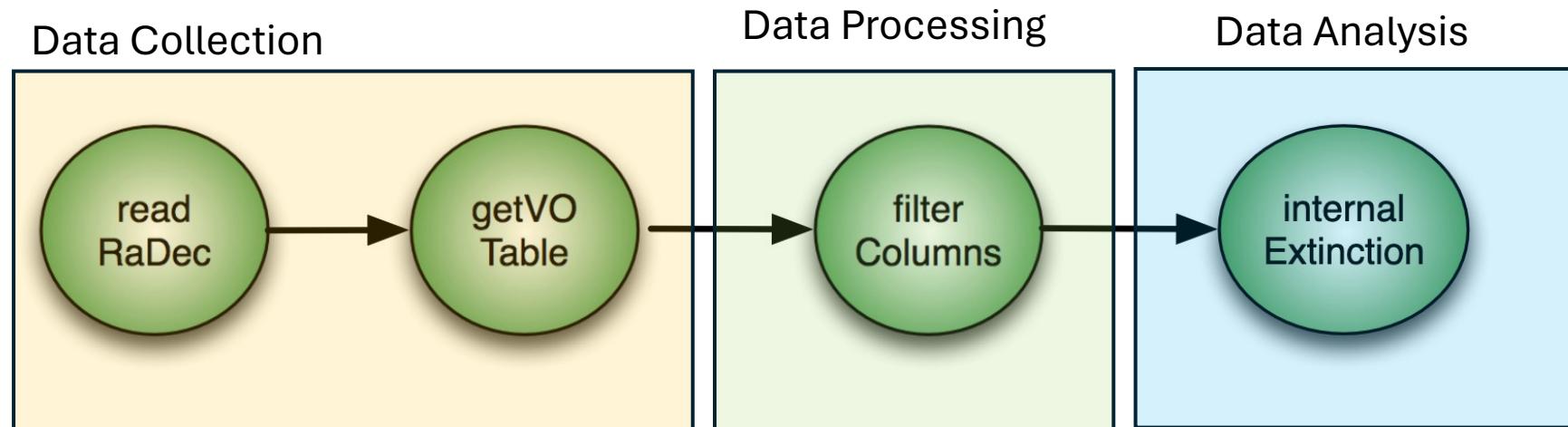
# Workflows Key Phases

- Seismological Example:
  - Quickly analyze earthquakes
  - Model the ground motion after earthquakes
  - Rapid assessment of earthquakes' impact, and emergency response



# Workflows Key Phases

- Astrophysics:
  - Calculating the Internal Extinction of Galaxies from the AMIGA catalogue
  - Crucial for determining a galaxy's optical luminosity.



# Workflow High-level Elements

- Task executions with dependencies
  - Specify a series of tasks to run
    - Data Collections, Data pre-processing and simulation
    - Data Analysis, Visualization, Storage
  - Outputs from one task may be inputs for another
    - DataFlow management – using streams/data between tasks
    - File management - using files between tasks
- Task scheduling
  - Some tasks may be able to run in parallel with other tasks
- Resource provisioning (getting processors)
  - Computational resources are needed to run jobs
- Ensuring accuracy and efficiency
  - Provenance generation – what is provenance in workflows ?

# Workflow High-Level Elements

- Provenance:

- Capturing information about the workflow
    - E.g. when a task was run ?; with which parameters?; what was the previous task ? And the next one ? ....
  - Enables reproducibility of results, transparency, data sharing

- Two types:

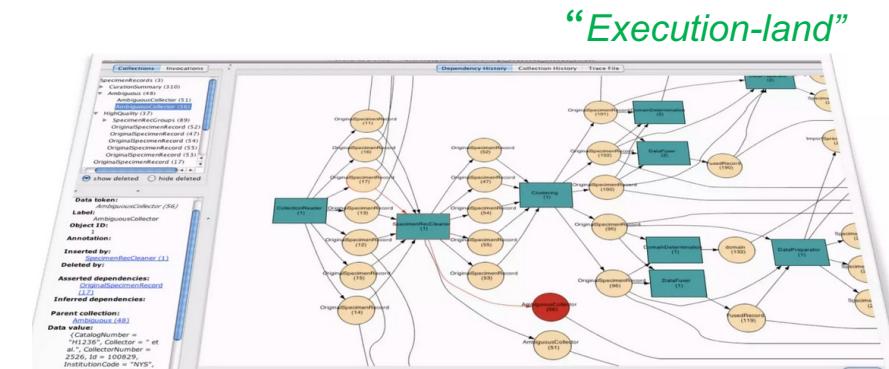
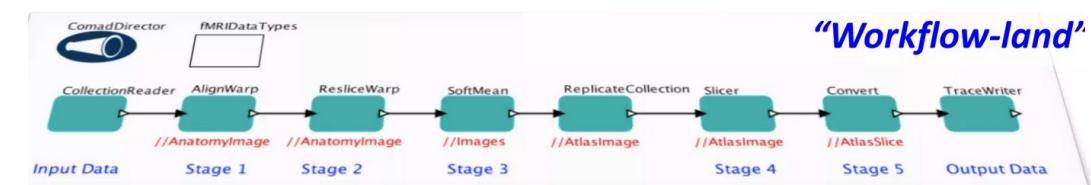
- Prospective provenance
  - Retrospective provenance

- Prospective provenance:

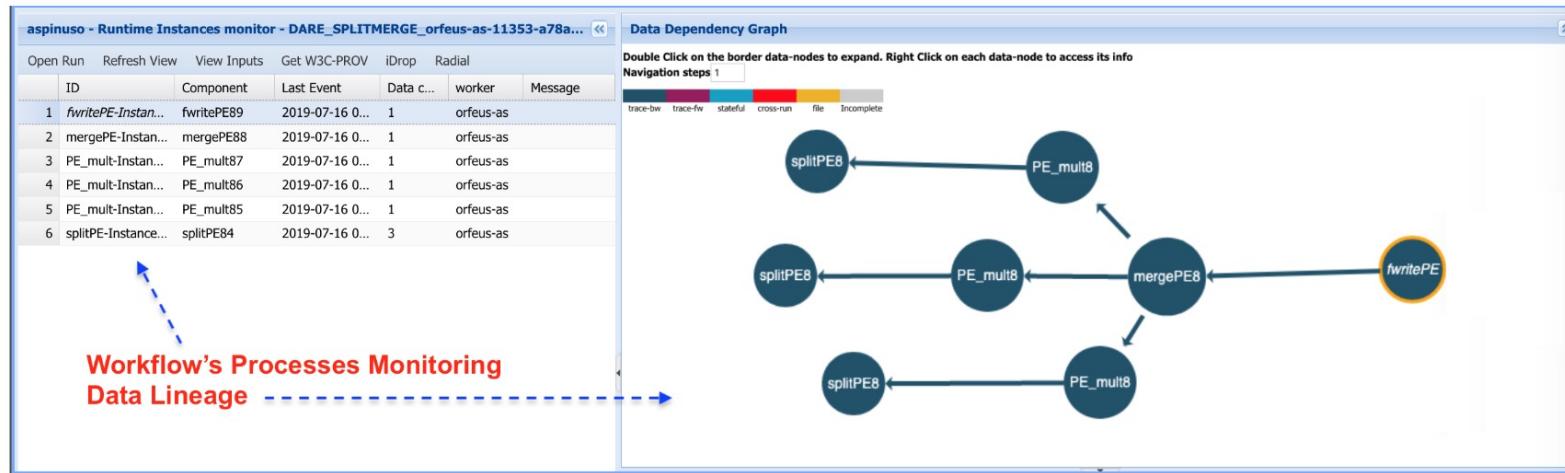
- captures the **structure and static context** of a workflow
  - it is independent of workflow execution or input data
  - provides an **abstract overview** of a workflow

- Retrospective provenance (~ tracing/profiling in HPC) :

- captures the **information about a workflow's execution**
  - becomes available when **running the workflow**.
  - facts about the execution of each workflow step & runtime environment



# Workflow High-Level Elements



[dispel4py provenance visualization of the execution of the Split and Merge workflow](#)

**retrospective provenance** → includes facts about the execution of each workflow step and the runtime environment

# What do we need help with?

- Task executions with dependencies
  - What if something fails in the middle?
  - Dependencies may be complex
- Task scheduling
  - Minimize execution time while preserving dependencies
  - May have many tasks to run
- Resource provisioning
  - May want to run across multiple systems
  - How to match processors to work?

# What do we need help with?

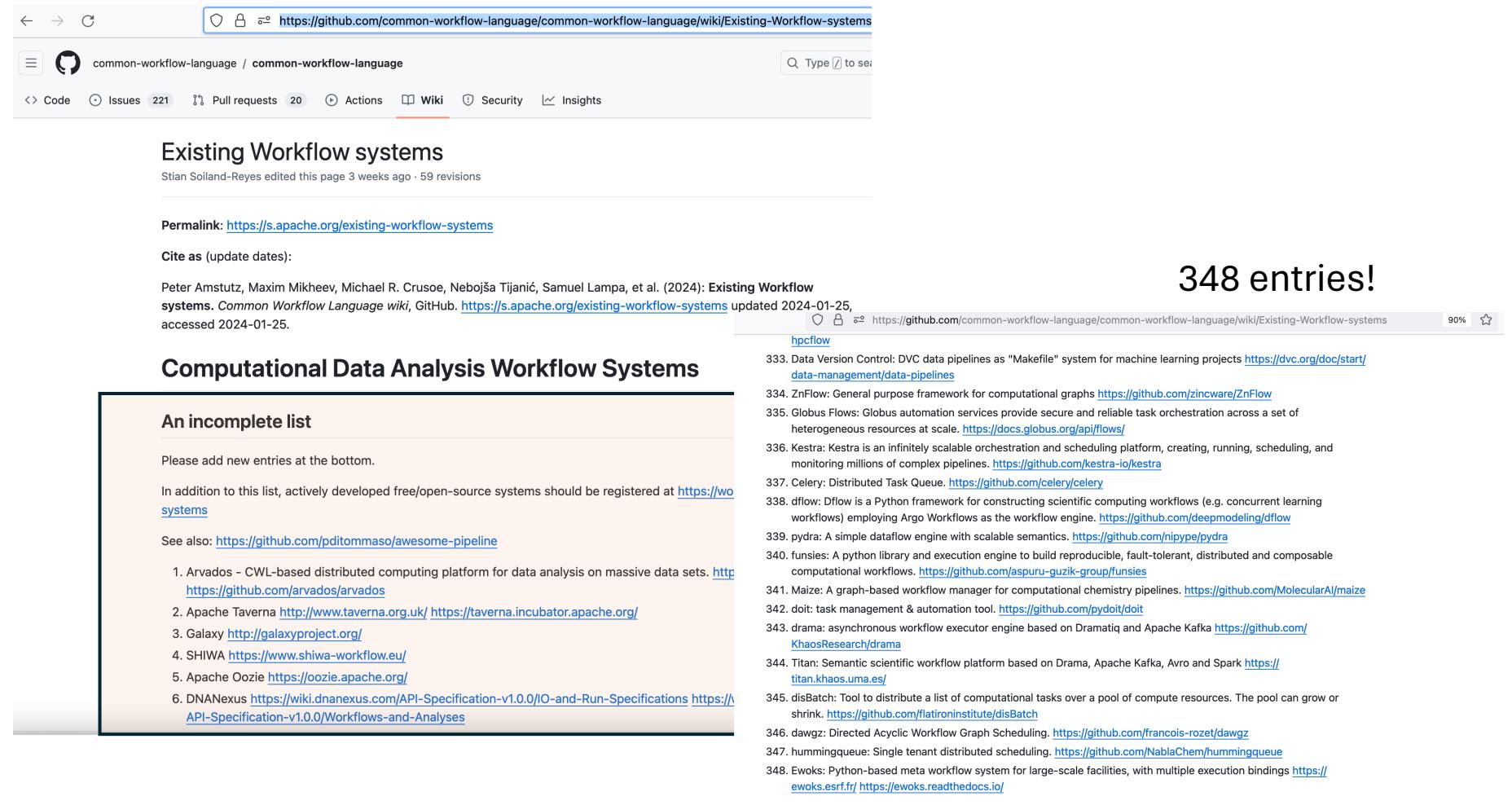
- Metadata and provenance
  - Automatically capture and track
  - Where did my task run? How long did it take?
  - What were the inputs and parameters?
  - What versions of code were used?
- File & Data management
  - Make sure inputs are available for tasks
  - Archive output data
- Automation
  - You have a workflow already – are there manual steps?

# Workflow Management Systems (WMS)

- Workflow Management Systems (WMS)
    - Software solutions that provide the infrastructure for
      - Designing
      - Executing
      - Monitoring scientific workflows
    - They offer tools for
      - automating tasks
      - managing data flow
      - ensuring that computational resources are used efficiently
  - Pegasus, Taverna, CWL, dispel4py, Apache Airflow, Nextflow, pycompss, ....
    - <https://github.com/common-workflow-language/common-workflow-language/wiki/Existing-Workflow-systems>
    - <https://workflows.community/systems>
    - [Bringing the Scientific Workflows Community Together](#)



# Workflow Management Systems (WMS)



The screenshot shows a GitHub wiki page titled "Existing Workflow systems". The page is part of the "common-workflow-language" repository. The URL is <https://github.com/common-workflow-language/common-workflow-language/wiki/Existing-Workflow-systems>. The page content includes a list of 348 entries, each with a title and a link. The first few entries are:

- hpcflow
- 333. Data Version Control: DVC data pipelines as "Makefile" system for machine learning projects <https://dvc.org/doc/start/data-management/data-pipelines>
- 334. ZnFlow: General purpose framework for computational graphs <https://github.com/zincware/ZnFlow>
- 335. Globus Flows: Globus automation services provide secure and reliable task orchestration across a set of heterogeneous resources at scale. <https://docs.globus.org/api/flows/>
- 336. Kestra: Kestra is an infinitely scalable orchestration and scheduling platform, creating, running, scheduling, and monitoring millions of complex pipelines. <https://github.com/kestra-io/kestra>
- 337. Celery: Distributed Task Queue. <https://github.com/celery/celery>
- 338. dflow: Dflow is a Python framework for constructing scientific computing workflows (e.g. concurrent learning workflows) employing Argo Workflows as the workflow engine. <https://github.com/deepmodeling/dflow>
- 339. pydra: A simple dataflow engine with scalable semantics. <https://github.com/nipype/pydra>
- 340. funsies: A python library and execution engine to build reproducible, fault-tolerant, distributed and composable computational workflows. <https://github.com/aspru-guzik-group/funsies>

The page also features a sidebar with sections like "Computational Data Analysis Workflow Systems" and "An incomplete list".

<https://github.com/common-workflow-language/common-workflow-language/wiki/Existing-Workflow-systems>

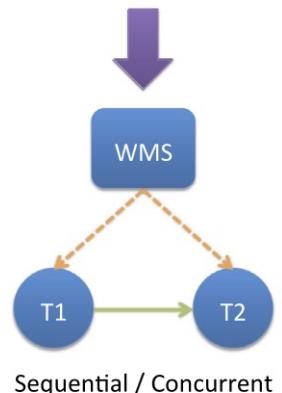
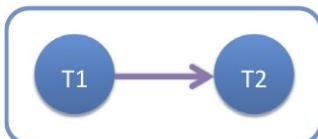
# Workflow Management Systems (WMS)

- Essential functions:
  - **Automate** programs and services scientists already use
  - **Support** one or several workflow execution modes.
  - **Schedule** invocation of programs and services correctly and efficiently – in **parallel** where possible.
  - **Manage dataflow** to, from and between programs and services
  - **Enable scientists** (no just developers) to author and modify workflow easily
- Desired functions:
  - **Predict** what a workflow will do when executed: **prospective provenance**.
  - **Record** what happened during workflow execution: **retrospective provenance**
  - **Reveal and query provenance** – how workflow products were derived from inputs
  - Enable scientists to **version, share and publish** their workflows

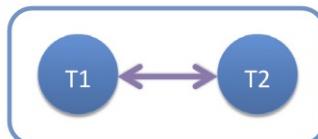
# Workflow Management Systems (WMS)

Workflow execution model

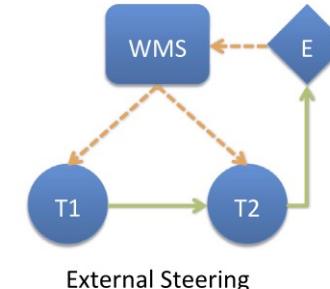
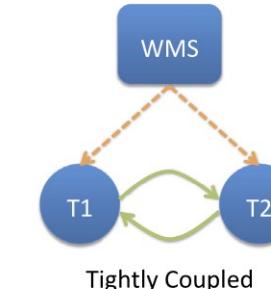
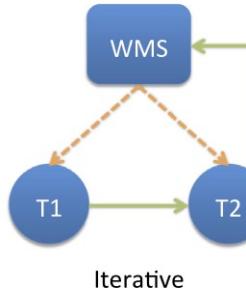
Acyclic Execution Model



Cyclic Execution Model

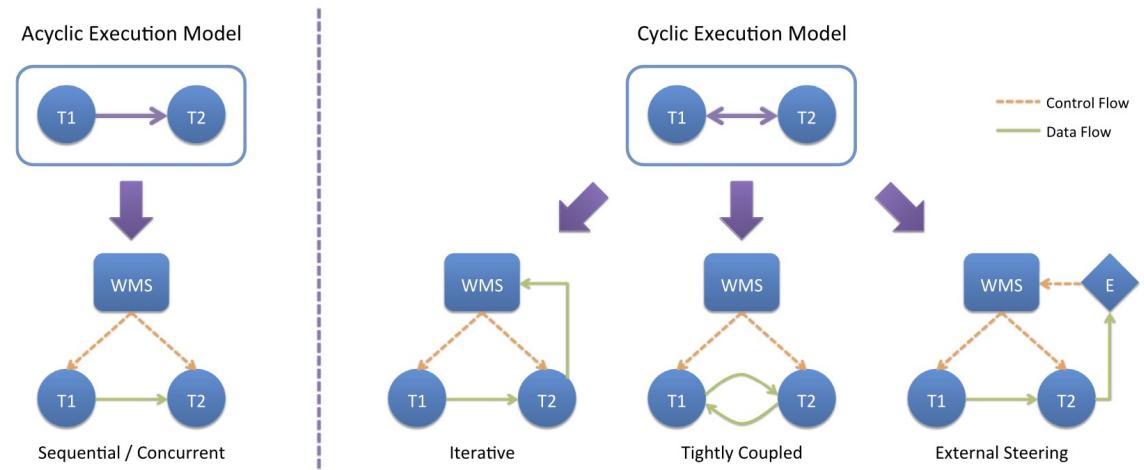


--> Control Flow  
--> Data Flow



# Workflow Management Systems (WMS)

- **Sequential:**
  - Begins with T1 followed by T2 upon completion;
  - Ideal for processes involving a simulation followed by post-processing.
- **Concurrent:**
  - T1 and T2 run simultaneously . T1's output feeding into T2;
  - Suitable for streaming data and multiple-stage processing
- **Iterative:**
  - Repeating sequences where T1's completion leads to T2, and T2's results may trigger further iterations
  - Useful for exploring parameter spaces through Uncertainty Quantification (UQ).
- **Tightly Coupled:**
  - T1 and T2 exchange partial results;
  - Supporting closely integrated simulations (e.g. multi-physics simulations that require tight coordination)
- **External Steering:**
  - Similar to concurrent but includes user/system evaluation of T2's output to potentially adjust T1 or other workflow aspects;
  - Useful for scenarios requiring real-time analysis feedback to guide subsequent steps



# 1.vi) Workflow Management Systems (WMS)

Classification of workflow management systems

Workflow Properties	ADIOS	Airavata	Askalon	Bobolang	dispel4py	Fireworks	Galaxy	Kepler	Makeflow	Moteur	Nextflow	Pegasus	Swift	Taverna	Triana
<i>Workflow Execution Models</i>															
Sequential	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Concurrent	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✓	✗
Iterative	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓	✓	✗
Tightly coupled	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
External steering	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
<i>Heterogeneous Computing Environments</i>															
Co-location	✗	✗	✗	✓	✓	✓	✗	✗	✗	✗	✓	✓	✓	✗	✗
External location	✗	✓	✓	✗	✗	✗	✓	✓	✓	✓	✗	✓	✓	✓	✓
<i>In situ</i>	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
<i>Data Access Methods</i>															
Memory	✓	✗	✗	✓	✓	✓	✗	✗	✗	✗	✓	✓	✓	✗	✗
Messages	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Local disk	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Shared file system	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Object store	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓	✗	✓	✓	✗
Other remote storage	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓	✓	✓	✓	✗

A characterization of workflow management systems for extreme-scale applications



## 2. How to select a WMS ?

# Key Factors to Consider

- Several WMS solve same general problems
  - but differ in specific approach
- A few categories to think :
  - Interface: how are workflows constructed?
  - Workload: what does your workflow look like?
  - Community: what domains does the tool focus on?
  - Task-Flow vs Data-Flow;

# Interface

- How does a user construct workflows?
  - Graphical: like assembling a flow chart
  - Scripting: use a workflow tool-specific scripting language to describe workflow
  - API: use a common programming language with a tool-provided API to describe workflow
- Which is best depends on your application
  - Graphical can be unwieldy with many tasks
  - Scripting and API can require more initial investment
  - Some WMS support multiple approaches

# Workload

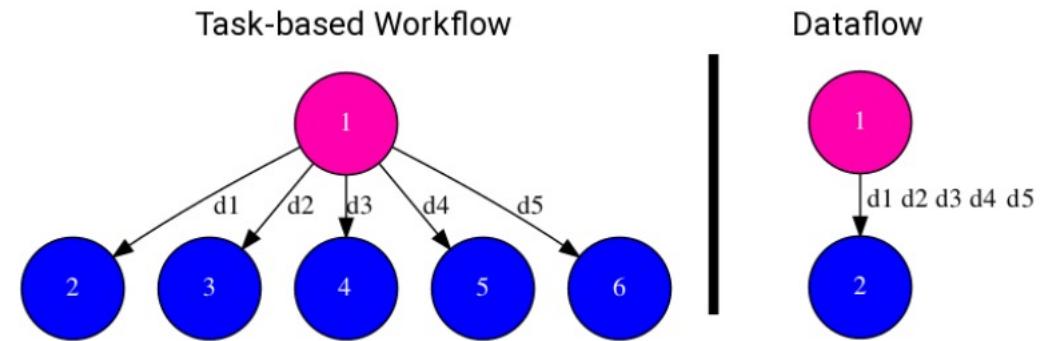
- What kind of workflow are you running?
  - Many vs. few tasks
  - Short vs. long
  - Dynamic vs. static
  - Loops vs. directed acyclic graph
- Different tools are targeted at different workload

# Community

- What kinds of applications is the tool designed for?
- Some tools focus on certain science fields
  - Have specific paradigms or task types built-in
  - Workflow community will share science field
  - Less useful if not in the field or users of the provided tasks
- Some tools are more general
  - Open-ended, flexible
  - Less domain-specific community

# Task-Flow vs Data-Flow

- Task-Flow:
  - Orchestrates isolated tasks processing data in sequence, based on data dependencies.
  - Ideal for structured, sequential processing where each task must complete before the next starts.
- Data-Flow:
  - Manages tasks for continuous data receipt and production;
  - Enables real-time data processing and inter-task communication without waiting for task completions.
- Key Difference:
  - Task-Flow focuses on sequential, dependent task execution, while Data-Flow emphasizes ongoing, real-time interaction between tasks



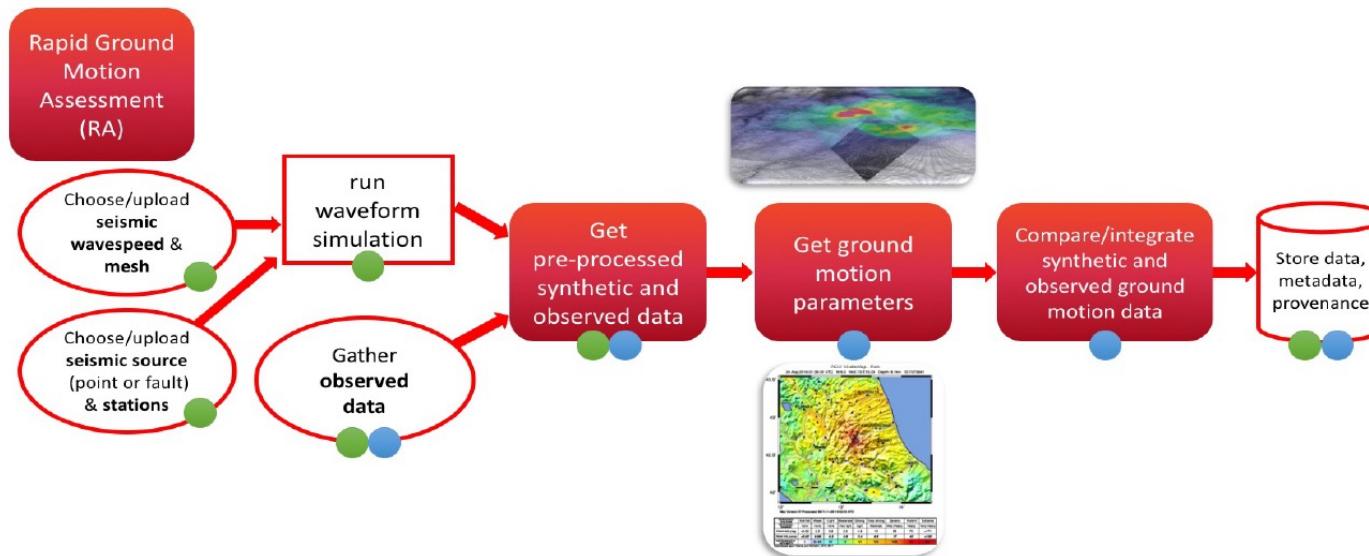
# Task-Flow vs Data-Flow



Common Workflow Language

# Real Examples – Rapid Assessment (RA)

- Rapid assessment (RA) of earthquakes' impact, and emergency response
  - Build dispel4py workflows to represent each part of the RA
    - Each part of RA follows a Data-Flow pattern
    - The full RA follows a Task-Flow pattern
  - Use CWL to connect RA dispel4py workflows



# Real Examples – Internal Extinction (IE)

- Calculating the Internal Extinction of Galaxies
  - Crucial for determining a galaxy's optical luminosity
  - Build the full workflow in dispel4py
  - Each task/step in IE follows a Data-Flow pattern

