



Getting a grip on the grid:

A knowledge base to trace grid experiments

Ammar Benabdelkader
ammarb@nikhef.nl

Mark Santcroos
m.a.santcroos@amc.uva.nl

Victor Guevara Masis
vguevara@nikhef.nl

Souley Madougou
souley@nikhef.nl

Antoine van Kampen
a.h.vankampen@amc.uva.nl

Silvia Olabarriaga
S.D.Olabarriaga@amc.uva.nl

BiG Grid

the dutch e-science grid



Presentation Outlines

- Background, challenges and Focus
- Provenance: an overview
- Provenance API (Plier):
 - Database schema
 - Architecture & Implementation
- eBioCrawler
 - Abstract/concrete graph
 - Challenges
- Plier Toolbox:
 - Generic functionalities
 - Customized functionalities
- Scientific Impact
- Conclusion & future work



BiG Grid

the dutch e-science grid





Big Grid (Dutch NGI)

- Founding partners: NCF, Nikhef and NBIC (2007-2011)
- Mission:

To realise a fully operational world-class and resources-rich grid environment at the national level in the Netherlands to serve public scientific research, including particle physics, life sciences and all other disciplines, and to encourage actively general grid usage across all disciplines.

- Details:
 - Ca. 25% for “user support” and “application-specific support”
 - Ca. 50% for “hardware infrastructure”
 - Ca. 25% for “running costs”
- Focus:
 - Grid: networking, compute, storage (resources) , databases, sensors, backup,
 - e-science: conducting science, using all kinds of ICT infrastructure and opportunities



BiG Grid

the dutch e-science grid





AMC: e-BioScience Group

- **Bioinformatics Laboratory**
 - Dept. Clinical Epidemiology, Biostatistics and Bioinformatics
 - Academic Medical Centre, University of Amsterdam
- Filling “gap” between medical researchers and the Dutch NGI
- Supporting a wide range of applications
 - Next Generation Sequencing
 - Medical Imaging
 - -Omics

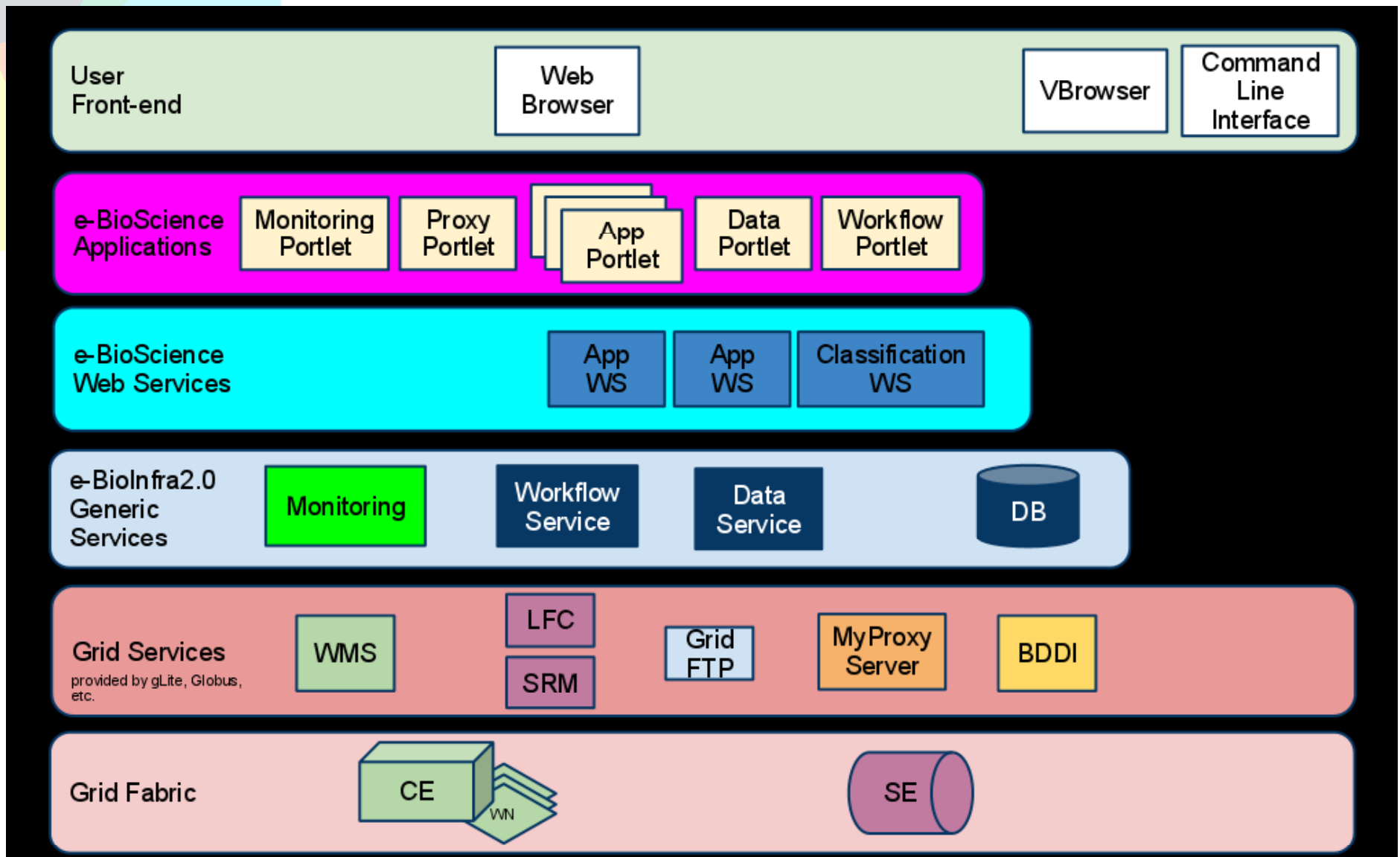


BiG Grid

the dutch e-science grid



e-BioScience Group: Layered Architecture



Background

- To run their experiment, e-BioScience group deploys:
 - **Moteur2/DIANE** Workflow engine, and
 - **GWENDIA** (Grid Workflow Efficient Enactment for Data Intensive Applications)
- Most experiments are complex due to:
 - Iteration over input parameters of running experiments
 - Each job is **instantiated** several times according to the number input data links.
 - **Re-trial** of failing process
 - Each failing job gets re-tried until it succeeded (or reaches re-trial limit)
 - Each workflow experiment may consist of a large number of failed and succeeded jobs.



BiG Grid

the dutch e-science grid





Challenges

- Hard to **validate** workflow experiments:
 - Identify whether an experiment **succeeded** or **failed**
 - Verify the **validity** of the output results
 - Identify the **source** of failure
- Hard to **instrument** and **document** experiments:
 - How to document validated experiments?
 - What to do with failed experiments?
 - How to keep track of the validation process?
 - How to preserve/publish the knowledge and expertise
- Hard to make **use of** the **gained** expertise:
 - How to prevent similar sources of failure?
 - How to spread the gained expertise?
 - How to better exploit the gained expertise?



BiG Grid

the dutch e-science grid



Focus

Build a **knowledge base** to instrument scientific experimentations

- Start with ...
 - **Building a knowledge** base to instruments scientific experimentations
 - Knowledge base should be flexible enough ...
- Adopt the **Open Provenance Model** (OPM) ...
 - Better suited to our case, since it provides history of occurrence of things (with flexibility)
 - Implement tools to build and store OPM-compliant data objects related to scientific experimentations
- Build **customized tools** to explore the data
- **Enhance** the database and Toolbox whenever needed.



BiG Grid

the dutch e-science grid





Open Provenance Model (1)

<http://openprovenance.org/>

- Allow us to express all the causes of an item
 - e.g., provenance of a scientific experiment includes:
 - Processes composing the experiment
 - Where did the processes run
 - What input they used
 - What results it generates, when and where
 - Who did launch and monitor the experiment
 - Etc.
- Allow for process-oriented and dataflow oriented views
- Based on a notion of annotated causality graph



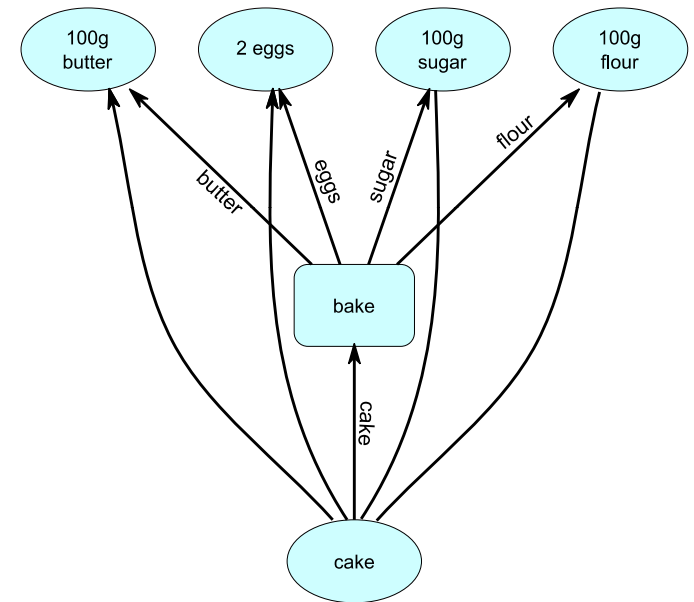
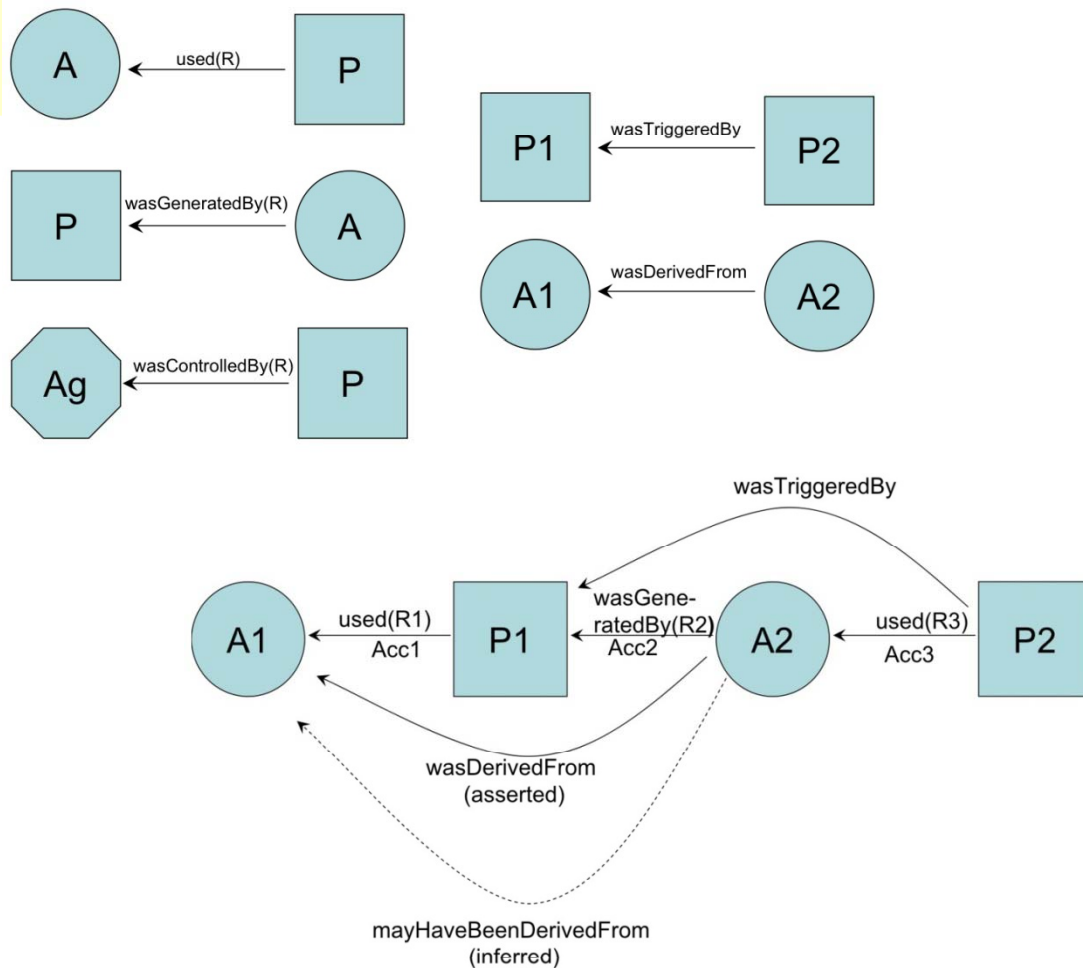
BiG Grid

the dutch e-science grid



Open Provenance Model (2)

<http://openprovenance.org/>



BiG Grid
the dutch e-science grid





PLIER Development

The *Provenance Layer Infrastructure for E-science Resources (PLIER)* provides an implementation of the Open Provenance Model (OPM)

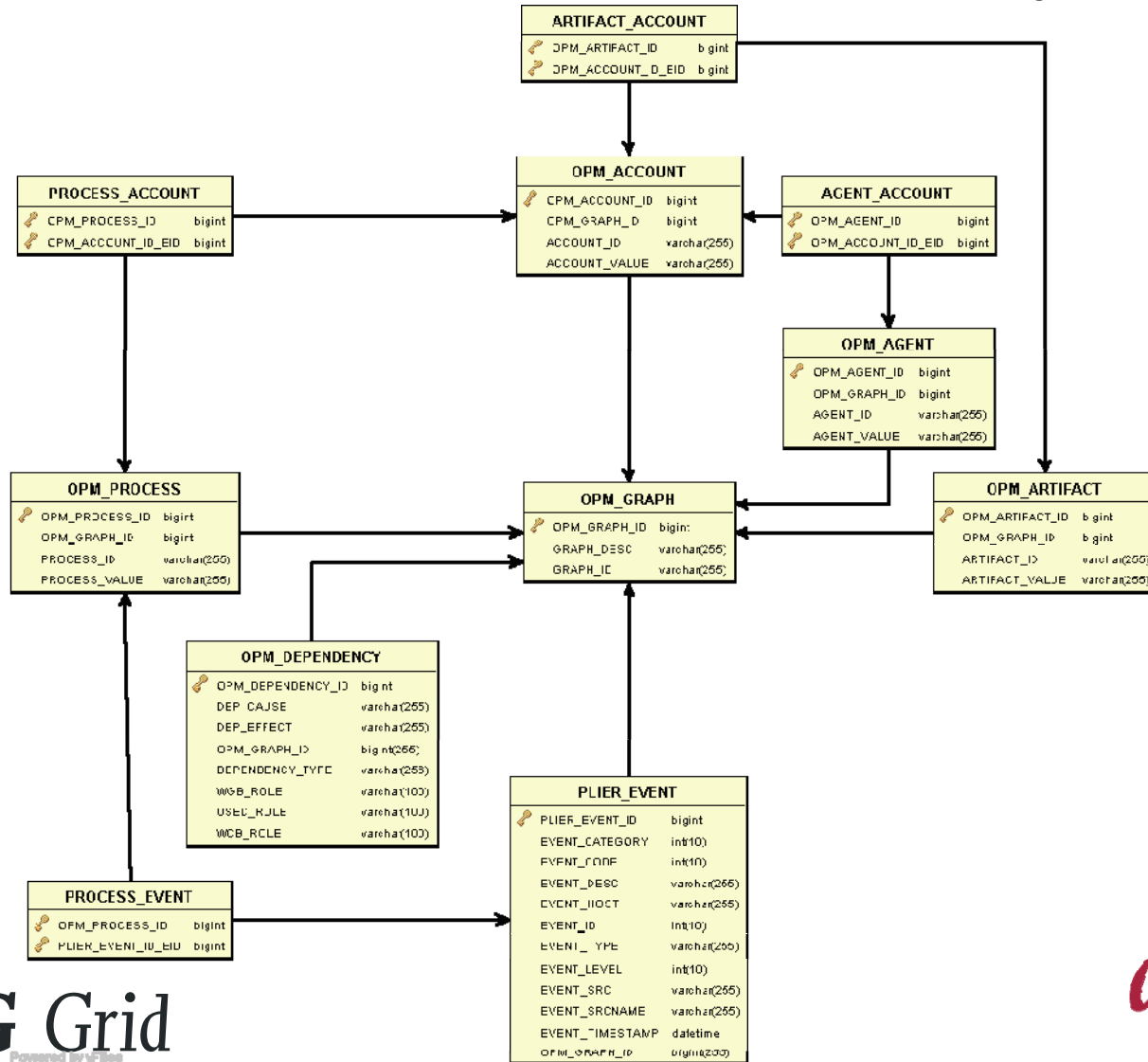
Four main components constitutes the Plier development:

1. Implementing the most optimum OPM-compliant relational database schema
2. Developing the Plier Core API:
Java-based API to **build** and **store** OPM graphs
3. Developing the eBioCrawler:
Java-based agents that **crawls** the input/output data for each experiments and **stores** it into the knowledge base.
4. Developing the Plier Toolbox:
Java-based UI to **visualize**, **search**, and **share** OPM graphs



PLIER: Database Schema

OPM compliant database schema used by Plier:





PLIER: Core API (1)

Plier API is implemented using most recent standards and mechanisms:

1. [JDO 3.1](#) is used as a java-centric API to access persistent data,
2. [DataNucleus](#) is used as a reference implementation of the JDO API,
3. [MySQL](#) is used as a back-end database to store provenance data

Plier Core API provides means to **build** OPM-compliant data objects and **store** them into the knowledge base





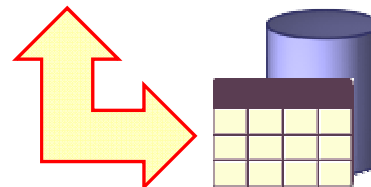
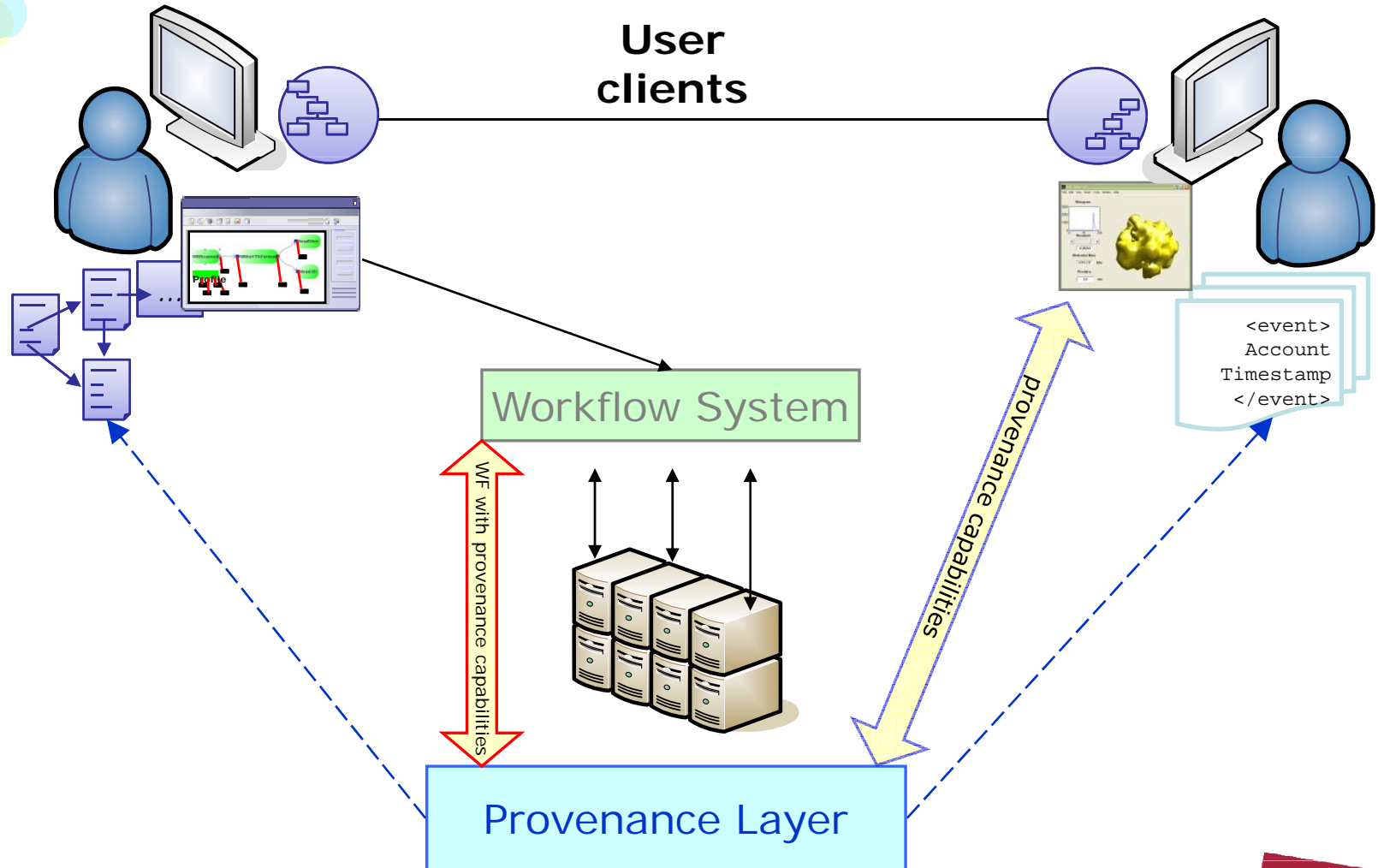
PLIER: Core API (2)

Plier API can be used in two manners:

1. Integrated within the workflow management system (WF with data provenance capabilities)
 - Scientists only need to enable the data provenance capabilities from the WF.
 - WF developers need to implement the DPC inside the workflow engine.
2. Implement the provenance data based on the input/output used/generated by the workflow system:
 - No need to change the workflow engine.
 - You may risk to build incomplete OPM graphs



PLIER: Core API (3)



eBioCrawler

Java-based agents that **crawls** the input/output data for each experiment and **stores** it into the knowledge base.

- Uses GWENDIA workflow description to build the **abstract model** of the experiment.
- Uses other **input/output/log** files to build the **concrete model** of the experiment.
- Workflow experiment data available through secure https server

→ **RISK:**

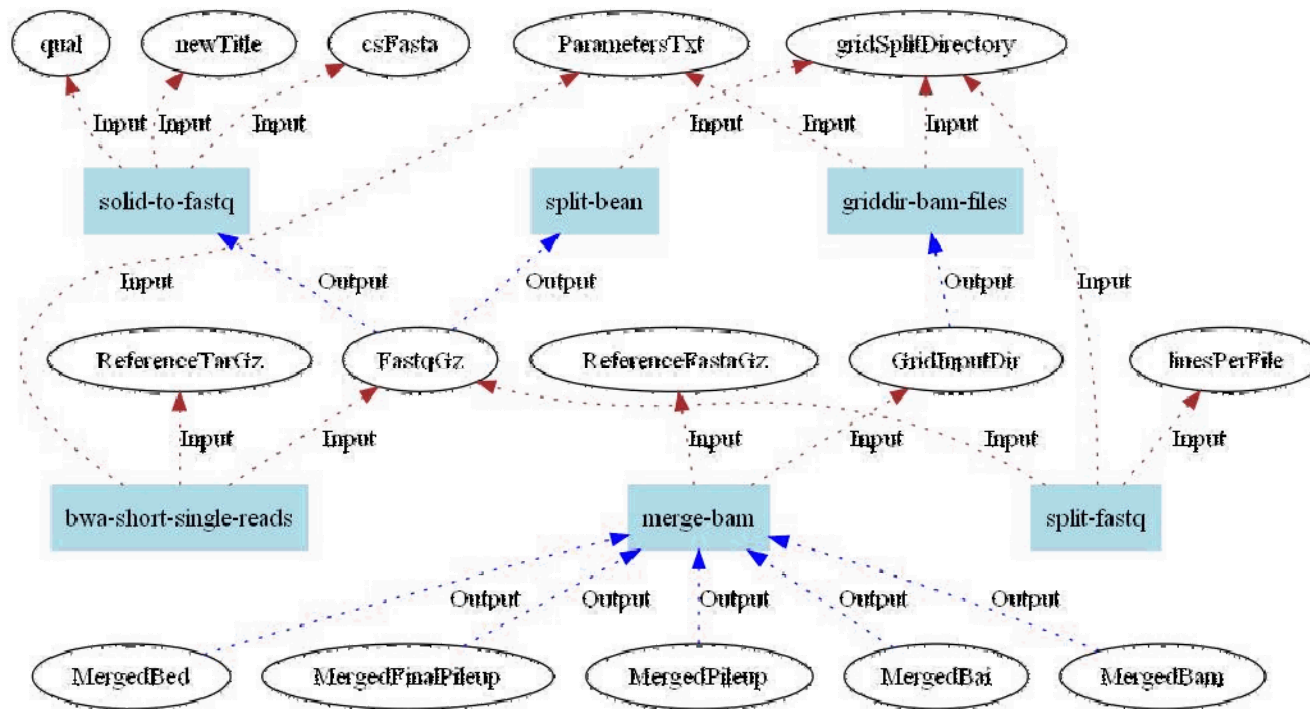
of not being able to collect/extract the required minimum data set of each experiment.



eBioCrawler: Abstract Graph

Extracted from the workflow description (GWENDIA XML format)

➔ Straight forward process



workflow-0b846ce3 (Abstract)





eBioCrawler: Concrete Graph

Extracted from the different **input/output/log** ,
used/generated by the workflow engine

 **complex process ...**

For each workflow experiments

- Users and host machines are modelled as **AGENTS**
- Executed Jobs are modelled as **PROCESSES**
- Input files/parameters are modelled as **ARTIFACTS**
- Output results are also modelled as **ARTIFACTS**
- Nodes are linked using **CAUSAL DEPENDENCIES**





eBioCrawler: Concrete Graph

Major issues, we faced:

- **Re-tried** processes causes data duplication, mainly with input files, which results in heavy graphs
- It was hard to identify **input files**/parameters for each job (values and order)
- **Output results** were hard to link to their corresponding processes

→ Most of the issues were solved by dedicating more programming efforts into eBioCrawler



BiG Grid
the dutch e-science grid





eBioCrawler: a success!

The approach was very successful:

- At first, eBioCrawler was able to collect about 70% of the required information
- With additional programming efforts into eBioCrawler we were able to collect more than 95% of the required information



This work is being used as a proof of concept to validate the suitability of the OPM model to our case



BiG Grid

the dutch e-science grid



PLIER Toolbox

Java-based UI to **visualize**, **search**, and **share** OPM graphs

1. Provide general functionalities like; **summary** of experiments with their **status**, **execution time**, etc.
2. Provide search functionalities based on keywords, user, date/time, status of experiment, etc.
3. Provide detailed information about each experiment/graph (e.g. input/output parameters, events, processes, etc.)
4. Provide OPMX (OPM XML format) and DOT (graphviz) data related to each experiment
5. Customized functionalities could be added to the interface (e.g. detailed report, analysis of output data, etc.)



BiG Grid

the dutch e-science grid



PLIER Toolbox- Search Menu

The screenshot shows the 'Provenance Query' window with the following fields and controls:

- Experiment**: Keyword
- Experimenter**: Name
- Timestamp**: Value
- Experiment status**: Status

Buttons on the right: Search, Reset, Settings, Metrics.

Table below the search fields:

Name	Stat		
workflow-0266fcb6	✓	90	03
workflow-027fb3ef	✗	00	00
workflow-03a7605h		nn	nn

Annotations with arrows pointing to the interface:

- (1) Keywords: Points to the 'Keyword' input field.
- (2) Timestamp: Points to the 'Value' input field.
- (3) Experimenter: Points to the 'Name' dropdown menu.
- (4) Exp. Status: Points to the 'Status' dropdown menu.



PLIER Toolbox- Exp. Summary

Exp. Status

- ✓ Total Success
- ✗ Total Failure
- ✓ Success with retry
- ✗ Partial Success

Experiment Duration

Experiment Summary

The screenshot shows the PLIER Toolbox interface. At the top, there are search and filter options: 'Experiment Keyword', 'Experimenter Name' (set to 'ALL'), and buttons for 'Search', 'Reset', 'Settings', and 'Metrics'. Below this is a table of experiments with columns for 'Name', 'Stat', and 'Time'. The 'Stat' column uses colored icons to indicate the status of each experiment.

Name	Stat	Time
workflow-0a144081	✓	04:23:05
workflow-0a206705	✓	00:00:34
workflow-0aa7e864	✗	00:00:00
workflow-0abb406c	✗	00:01:34
workflow-0ae76ee7	✗	00:00:00
workflow-0aee6bc2	✓	00:32:11
workflow-0afb0eee	✓	00:01:15
workflow-0b3ec9d6	✗	00:00:00
workflow-0b8b1f08	✗	00:00:00
workflow-0b8f0d89	✗	00:00:00
workflow-0b38e5ec	✗	00:00:00
workflow-0b846ec3	✗	01:34:58
workflow-0ba9d1a6	✗	03:54:22
workflow-0bbe0e10	✓	04:10:50
workflow-0bc89686	✗	00:01:31
workflow-0be73d99	✗	21:46:35
workflow-0be3861c	✗	03:04:05
workflow-0bfd7a	✓	
workflow-0c3d3d49	✗	
workflow-0c4a7b25	✗	
workflow-0c4e53de	✓	
workflow-0c5c5a80	✗	
workflow-0c6a2fb0	✗	
workflow-0c6ed3d6	✓	
workflow-0c8fff1b	✓	
workflow-0c81a9ed	✗	
workflow-0c25784d	✗	
workflow-0c78729b	✗	00:00:00
workflow-0cd42f9a	✓	22:22:12
workflow-0cf885bd	✓	00:00:17
workflow-0d08eeb0	✓	00:48:06
workflow-0d2fbbd6	✗	00:04:08
workflow-0d87a011	✗	00:00:00
workflow-0d275bd8	✗	00:00:00
workflow-0d1118fa	✗	00:00:00
workflow-0d702450	✗	00:00:00
workflow-0dcd2ac4	✓	00:01:07
workflow-0dd506da	✓	01:03:40
workflow-0deeb6e8	✗	00:00:00

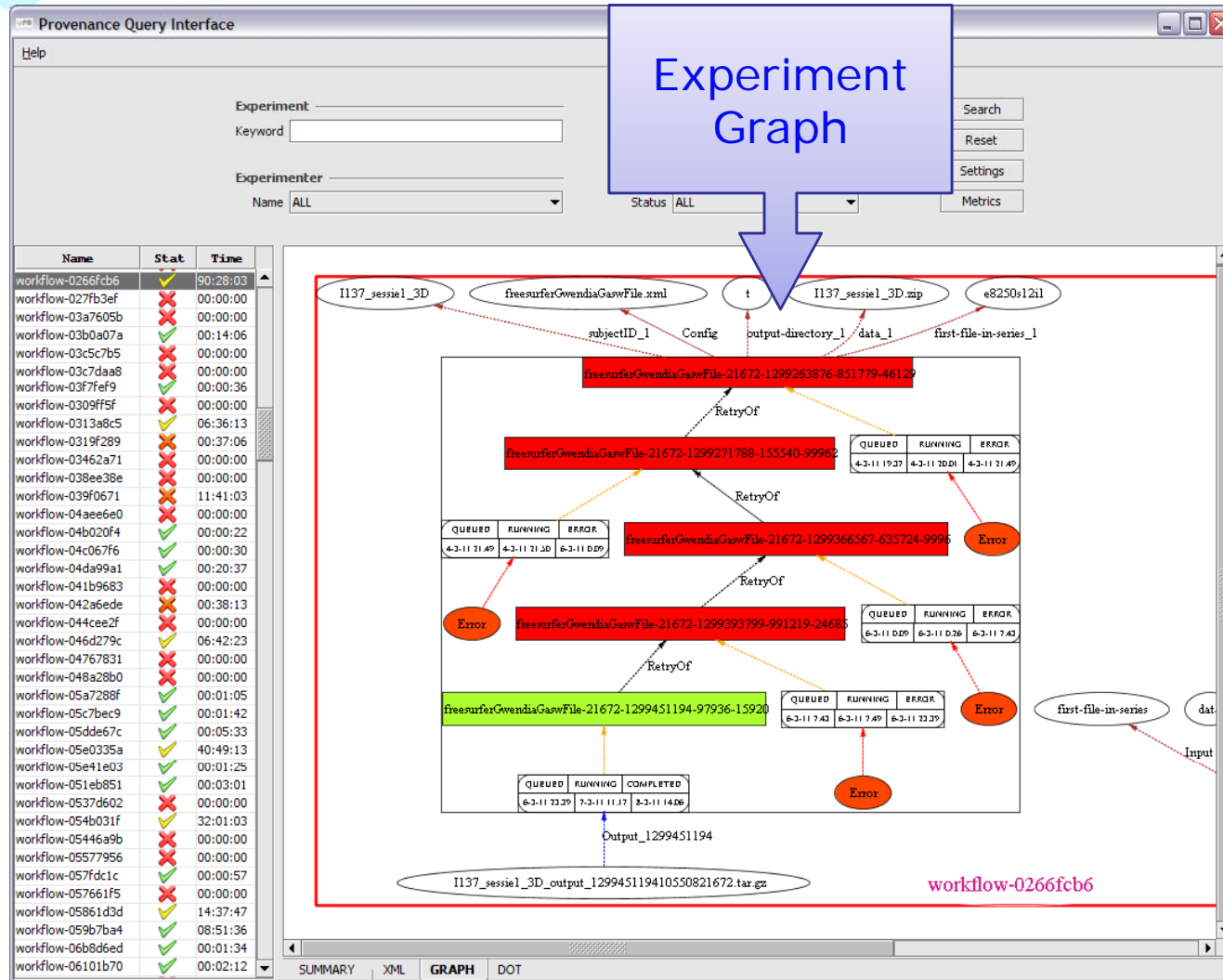
The right-hand pane shows a detailed summary for the selected experiment 'workflow-0a144081'. It includes an 'OPM Graph' (not visible), a 'Summary' table, and a 'Details' table.

Start	Duration	events	Parameters	Processes
2011-03-18 00:56:34.0	04:23:05	18	33	11

Process Id	Value
pileup	pileup
varscan-pileup-to-snp	varscan-pileup-to-snp
varscan-pileup-to-indel	varscan-pileup-to-indel
varscan-to-annovar	varscan-to-annovar
annovar-summarize	annovar-summarize
Pileup-7948-1300406192-418599-35719.sh	Pileup-7948-1300406192-418599-35719
Pileup-7948-1300408384-11085-11732.sh	Pileup-7948-1300408384-11085-11732
48-1300412678-912171-74084.sh	VarscanPileup2Snp-7948-1300412678-912171-74084
948-1300412685-896608-70525.sh	VarscanPileup2Indel-7948-1300412685-896608-70525
8-1300420374-553523-9252.sh	Varscan2Annovar-7948-1300420374-553523-9252
48-1300420442-970818-15509.sh	AnnovarSummarize-7948-1300420442-970818-15509



PLIER Toolbox – Exp. Graph



PLIER Toolbox – Exp. DOT

Provenance Query Interface

Experiment: _____
 Keyword: _____
 Experimenter: Name ALL Status _____

Search
 Reset
 Settings
 Metrics

Name	Stat	Time
workflow-0a144081	✓	04:23:05
workflow-0a206705	✗	00:00:34
workflow-0aa7e864	✗	00:00:00
workflow-0abb406c	✗	00:01:34
workflow-0ae76ee7	✗	00:00:00
workflow-0aee6bc2	✓	00:32:11
workflow-0afb0eee	✓	00:01:15
workflow-0b3ec9d6	✗	00:00:00
workflow-0bb1f08	✗	00:00:00
workflow-0b8f0d89	✗	00:00:00
workflow-0b38e5ec	✗	00:00:00
workflow-0b846ce3	✗	01:34:58
workflow-0ba9d1a6	✗	03:54:22
workflow-0bbe0e10	✓	04:10:50
workflow-0bc9e866	✓	00:01:31
workflow-0be73d99	✗	21:46:35
workflow-0be3861c	✓	03:04:05
workflow-0bfedf7a	✓	03:12:53
workflow-0c3d3d49	✗	00:00:00
workflow-0c4a7b25	✗	01:30:55
workflow-0c4e53de	✗	00:01:23
workflow-0c5c5a80	✓	20:47:01
workflow-0c6a2fb0	✗	20:47:00
workflow-0c6ed3d6	✓	01:57:34
workflow-0c8ff1b	✓	00:00:58
workflow-0c81a9ed	✗	00:00:00
workflow-0c25784d	✗	00:00:00
workflow-0c78729b	✗	00:00:00
workflow-0cd42f9a	✓	22:22:12
workflow-0cf095bd	✓	00:00:17
workflow-0d08eeb0	✓	00:04:08
workflow-0d27bbd6	✗	00:04:08
workflow-0d87a011	✗	00:00:00
workflow-0d275bd8	✗	00:00:00
workflow-0d1118fa	✗	00:00:00
workflow-0d702450	✗	00:00:00
workflow-0dc22ac4	✓	00:01:07
workflow-0dd506da	✓	01:03:40
workflow-0deeb668	✗	00:00:00

```

digraph OPMGraph {
  rankdir="BT";
  ratio=fill;fontsize=20;fontcolor=violetred;
  workflowLabel [label="workflow-0be3861c",color=white,fontsize=24;fontcolor=violetred,URL="https://orange.ebioscience.com/compound=true"];
  _pileup [shape=box,color=lightblue,style=filled,label="pileup"];
  _varscan_pileup_to_snp [shape=box,color=lightblue,style=filled,label="varscan-pileup-to-snp"];
  _varscan_pileup_to_indel [shape=box,color=lightblue,style=filled,label="varscan-pileup-to-indel"];
  _varscan_to_annovar [shape=box,color=lightblue,style=filled,label="varscan-to-annovar"];
  annovar_summarize [shape=box,color=lightblue,style=filled,label="annovar-summarize"];
  ReferenceFastaGz [shape=ellipse,label="ReferenceFastaGz"];
  BamFile [shape=ellipse,label="BamFile"];
  PileupFile [shape=ellipse,label="PileupFile"];
  VarscanParametersTxt [shape=ellipse,label="VarscanParametersTxt"];
  IndelFile [shape=ellipse,label="IndelFile"];
  SnpFile [shape=ellipse,label="SnpFile"];
  Varscan_INDEL [shape=ellipse,label="Varscan_INDEL"];
  Varscan_SNP [shape=ellipse,label="Varscan_SNP"];
  AnnovarFile [shape=ellipse,label="AnnovarFile"];
  HgX_humandbTarGz [shape=ellipse,label="HgX_humandbTarGz"];
  AnnovarSummaryTarGz [shape=ellipse,label="AnnovarSummaryTarGz"];
  ExomeCsv [shape=ellipse,label="ExomeCsv"];
  GenomeCsv [shape=ellipse,label="GenomeCsv"];
  ReferenceFastaGz_1 [shape=ellipse,label="hg19_all.fasta.gz"];
  BamFile_1 [shape=ellipse,label="run19_sample5.bam"];
  VarscanParametersTxt_1 [shape=ellipse,label="hg19_humandb.tar.gz"];
  Pileup_xml [shape=ellipse,label="Pileup.xml"];
  VarscanPileup2Snp_xml [shape=ellipse,label="VarscanPileup2Snp.xml"];
  run19_sample5_bam_f_pileup_gz [shape=ellipse,label="run19_sample5.bam.f.pileup.gz"];
  VarscanPileup2Indel_xml [shape=ellipse,label="VarscanPileup2Indel.xml"];
  PileupFile_1 [shape=ellipse,label="run19_sample5.bam.f.pileup.gz"];
  SnpFile_1 [shape=ellipse,label="run19_sample5.bam.f.pileup.gz.snp"];
  IndelFile_1 [shape=ellipse,label=""];
  AnnovarFile_1 [shape=ellipse,label=""];
  AnnovarSummaryTarGz_1 [shape=ellipse,label=""];
  ExomeCsv_1 [shape=ellipse,label=""];
  GenomeCsv_1 [shape=ellipse,label=""];
  User_Aldo_Jongejan [shape=octagon,color=lightcyan,style=filled,label="Aldo Jongejan"];

  _pileup -> _ReferenceFastaGz [style=dotted,color=brown,label="Input"];
  _pileup -> BamFile [style=dotted,color=brown,label="Input"];
  }
  
```

SUMMARY XML GRAPH DOT

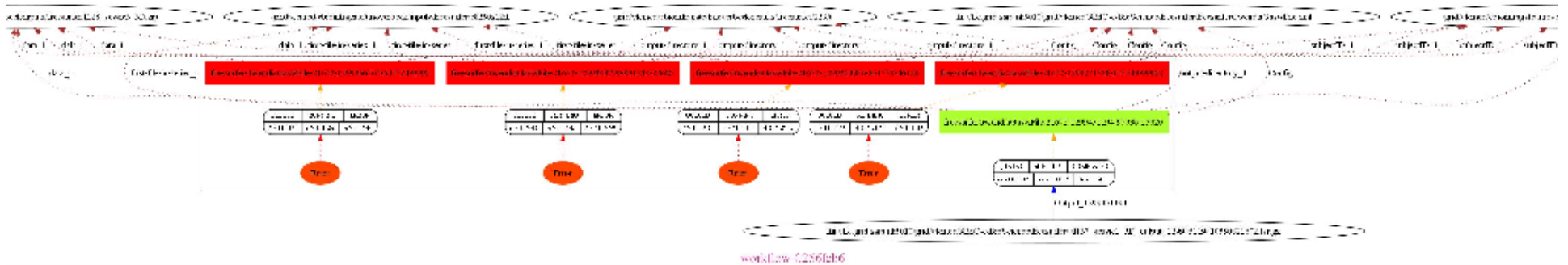
Experiment
DOT



Graph Customization

Initial Graph

Long path names



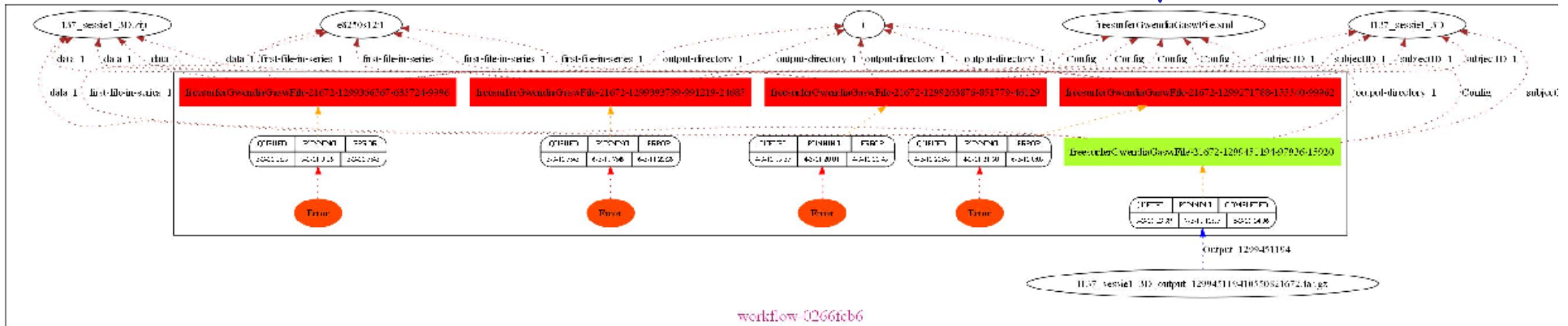
BiG Grid
the dutch e-science grid



Graph Customization

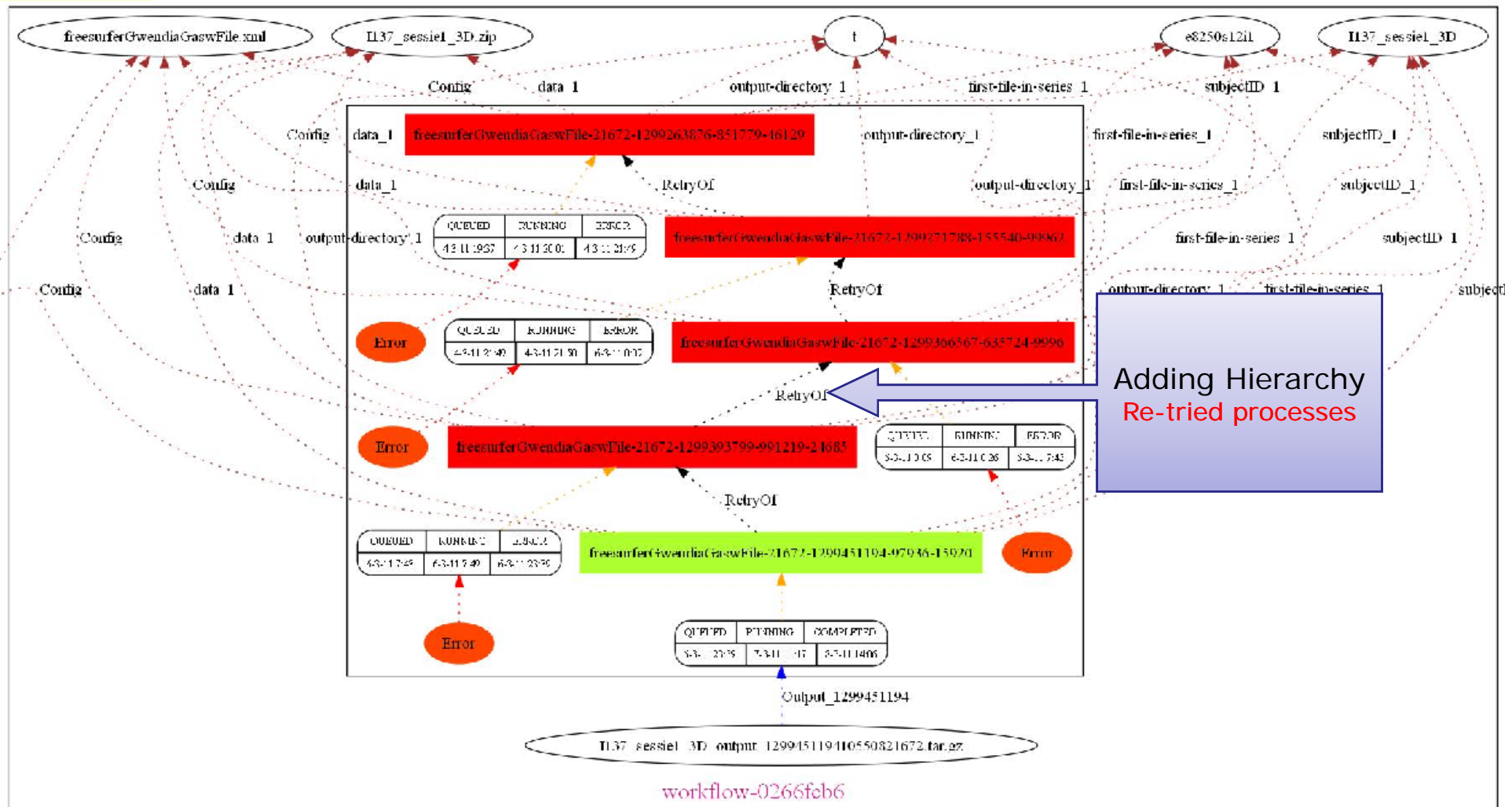
Hiding artifact Path

Hiding the path name



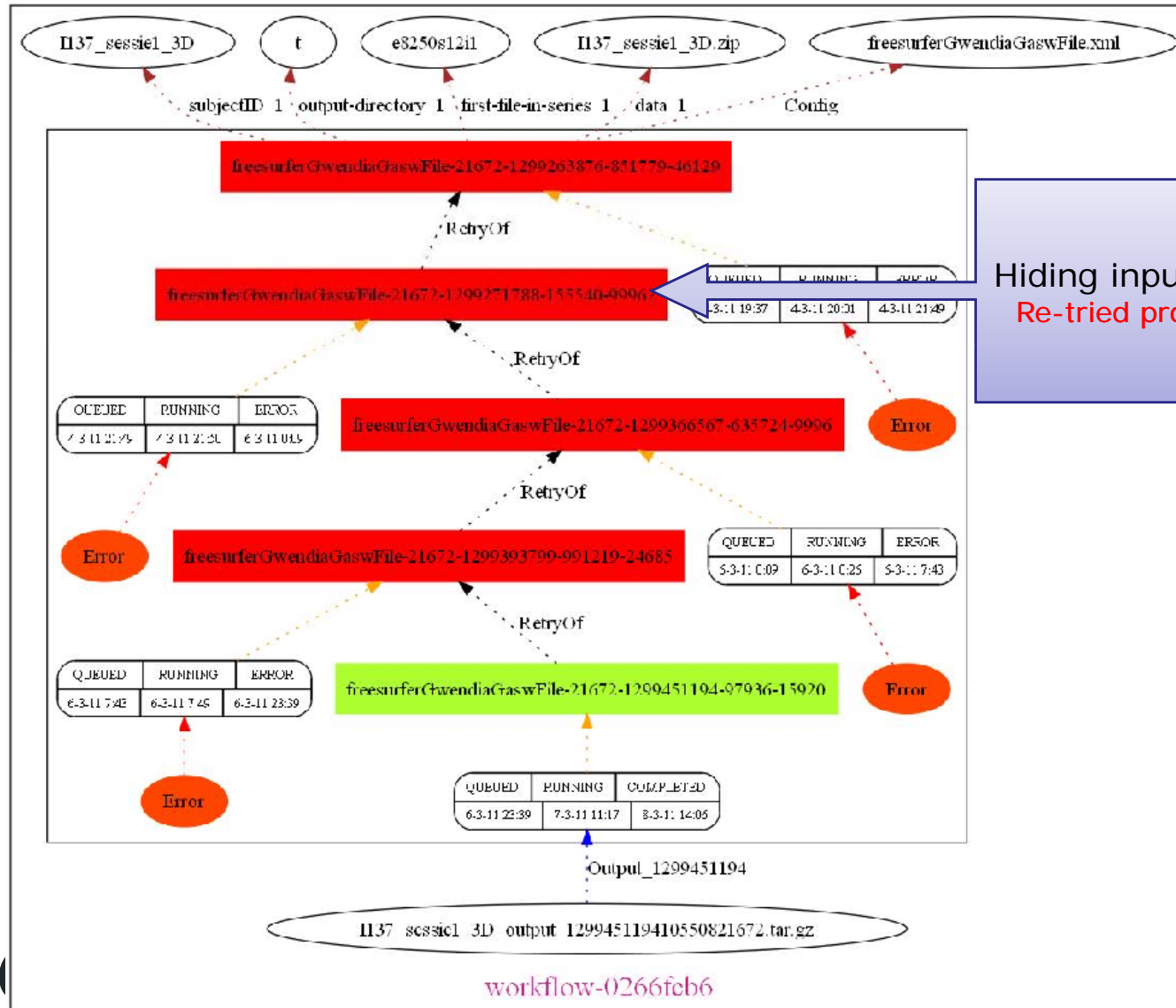
Graph Customization

Adding process hierarchy



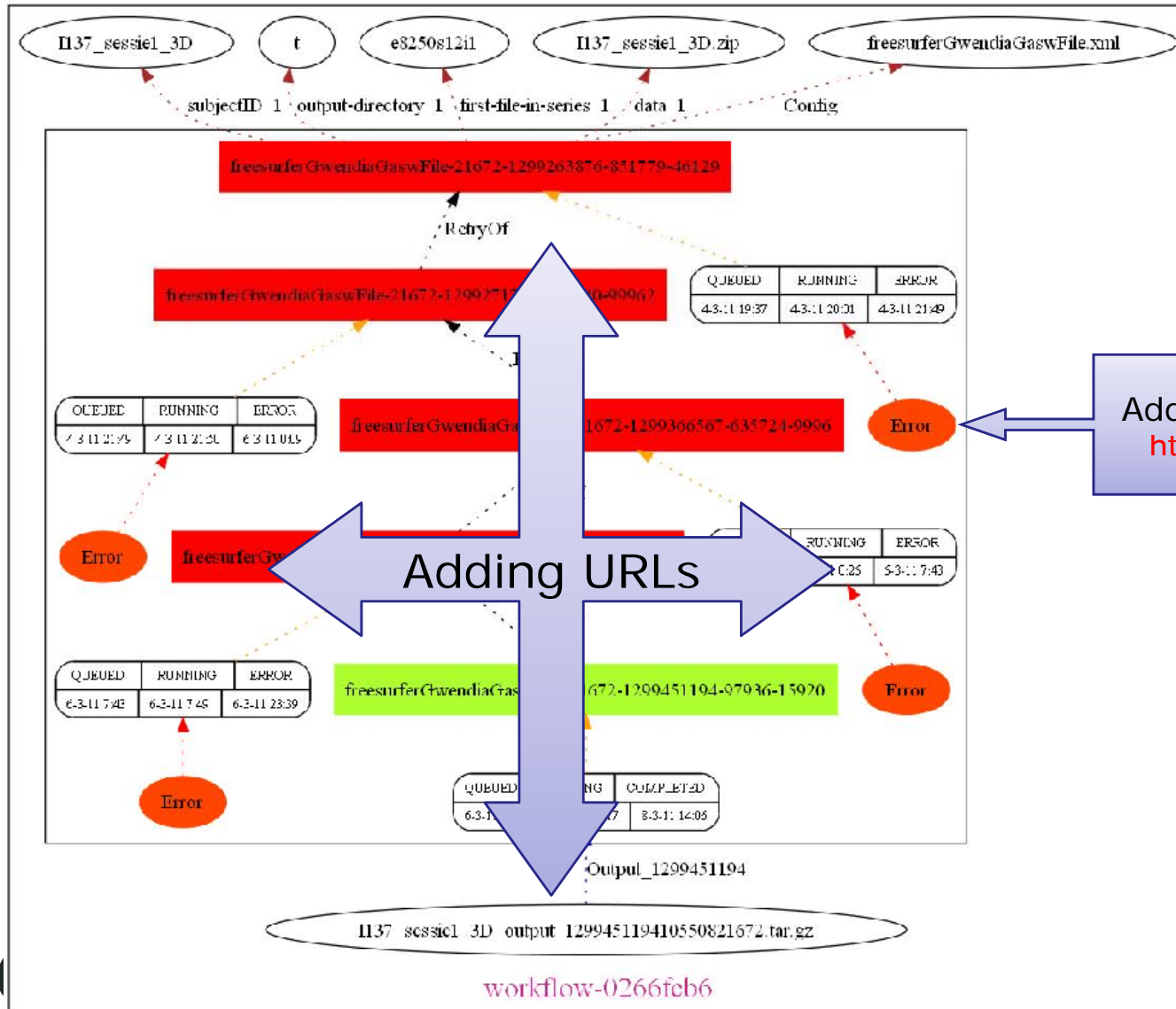
Graph Customization

Hiding duplicate inputs



Graph Customization

Adding URL Links





Conclusion and Future Work

Usefulness - we were able to :

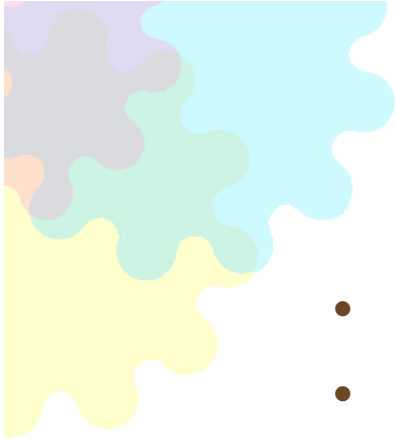
- identify **final status** of experiments (5 status)
- easily **trace** the source of error
- identify the **reason** for error occurrence
- **decide** what to do with failing jobs
- **clean** the grid storage (outputs of failing exp.)
- Etc.



BiG Grid

the dutch e-science grid





Conclusion and Future Work

- Enhance Plier API to OPM core specifications (v1.1)
- Implement the provenance model into the Moteur workflow engine
- Enhance the data management Toolbox with additional components:
 - Improve the search criteria
 - Documenting, annotating, reviewing and publishing experiments
 - Fully automate the process of validating experiments
- Extend the usage of the data management Toolbox to other groups





Acknowledgments

- **Big Grid**

This work is part of the program of Big Grid, the Dutch e-Science Grid, which is financially supported by the Netherlands Organization for Scientific Research, NWO.

- **AMC**



E-BioScience Group,

- **Modalis Team**



Developers of MOTEUR WS



BiG Grid

the dutch e-science grid





Useful Links

- Plier Core API:
<http://twiki.ipaw.info/bin/view/OPM/Plier>
- Plier Toolbox:
<http://twiki.ipaw.info/bin/view/OPM/PlierToolBox>
- eBioCrawler:
<http://bioinformaticslaboratory.nl/twiki/bin/view/EBioScience/EBioCrawler>
- Open Provenance Model (OPM):
<http://openprovenance.org/>
- Moteur:
<http://modalis.i3s.unice.fr/moteur2>
- DIANE:
<http://it-proj-diane.web.cern.ch/it-proj-diane/>



BiG Grid

the dutch e-science grid

