Automatic Generation of Portals for Distributed Scientific Applications

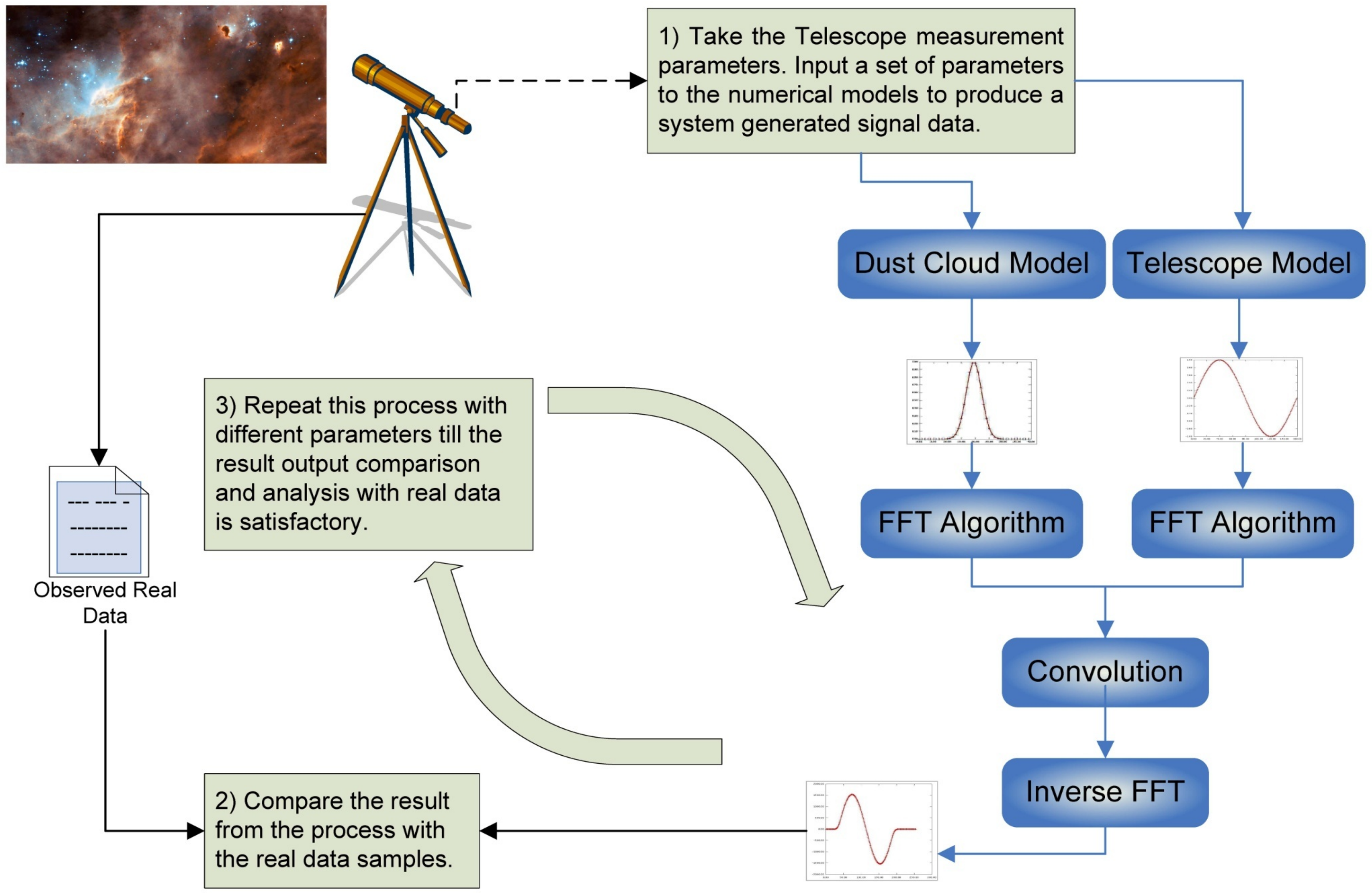
David W. Walker and Dashan Lu
School of Computer Science and Informatics
Cardiff University, UK
david@cs.cf.ac.uk

Overview

- Portals to distributed scientific applications
- Introduction and motivation for the automatic portal creation.
- Portal building based on XML workflow descriptions.
- Embedding interfaces in published documents.

Application Portals

- Application portals are web-accessible interfaces to distributed applications.
- These applications can be expressed as workflows, represented by DAGs, and expressed in XML.
- DAG nodes are Web services, and arcs are SOAP messages.
- Node tasks may run concurrently, and may contain internal parallelism that can be exploited through MPI, OpenMP, MapReduce, etc.
- Hide complexities of using distributed resource from the users



1) Take the Telescope measurement parameters. Input a set of parameters to the numerical models to produce a system generated signal data.

2) Compare the result from the process with the real data samples.

3) Repeat this process with different parameters till the result output comparison and analysis with real data is satisfactory.

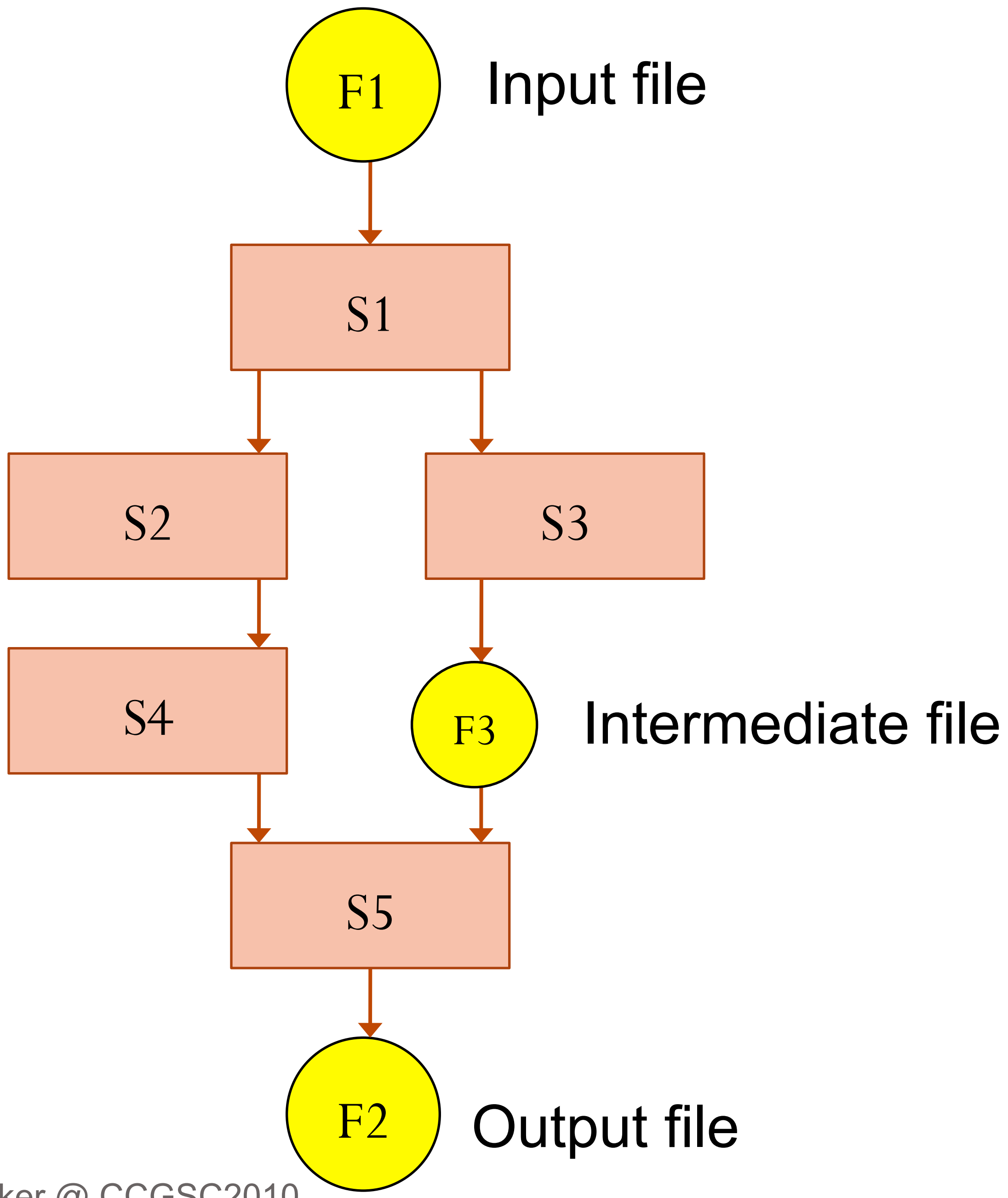
Motivation

- Scientific application users often want to:
  - Replace one algorithm with another
  - Replace one file with another

  while the structure of workflow stays the same.

OR

- They may want to edit an existing workflow, for example, to insert a new node.
- Basic entities in the workflow are task nodes (services) and data nodes (files).


F1
Input file
S1
S2
S3
S4
F3
Intermediate file
S5
F2
Output file

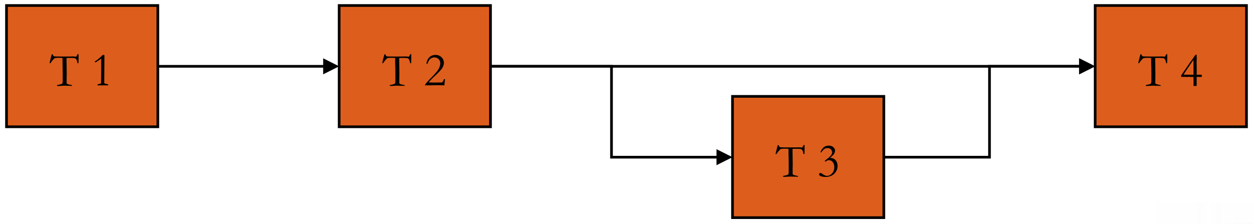
Different Types of Workflow

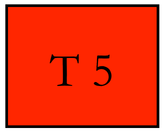
- A skeleton workflow is a DAG composed of task and data nodes, without specifying any details.
- An abstract workflow specifies what the services that make up the workflow do, but does not specify particular implementations for these services. This is an application template.
- A concrete workflow specifies not only what the services in the workflow do, but also specifies their service implementations. Specific files are also identified. A concrete workflow is executable.

What a Portal Does

- For this type of application we need a portal to provide an interface through which a user can:
  - Associate a task node with a particular service implementation. The user selects between semantically equivalent services.
  - Associate a data node with a particular file.
  - Execute the workflow.
- Thus the portal converts an abstract workflow into a concrete workflow.
- The above tasks are carried out by the portal using JSR-168 portlets.

Restrictions of Hand-crafted Portals



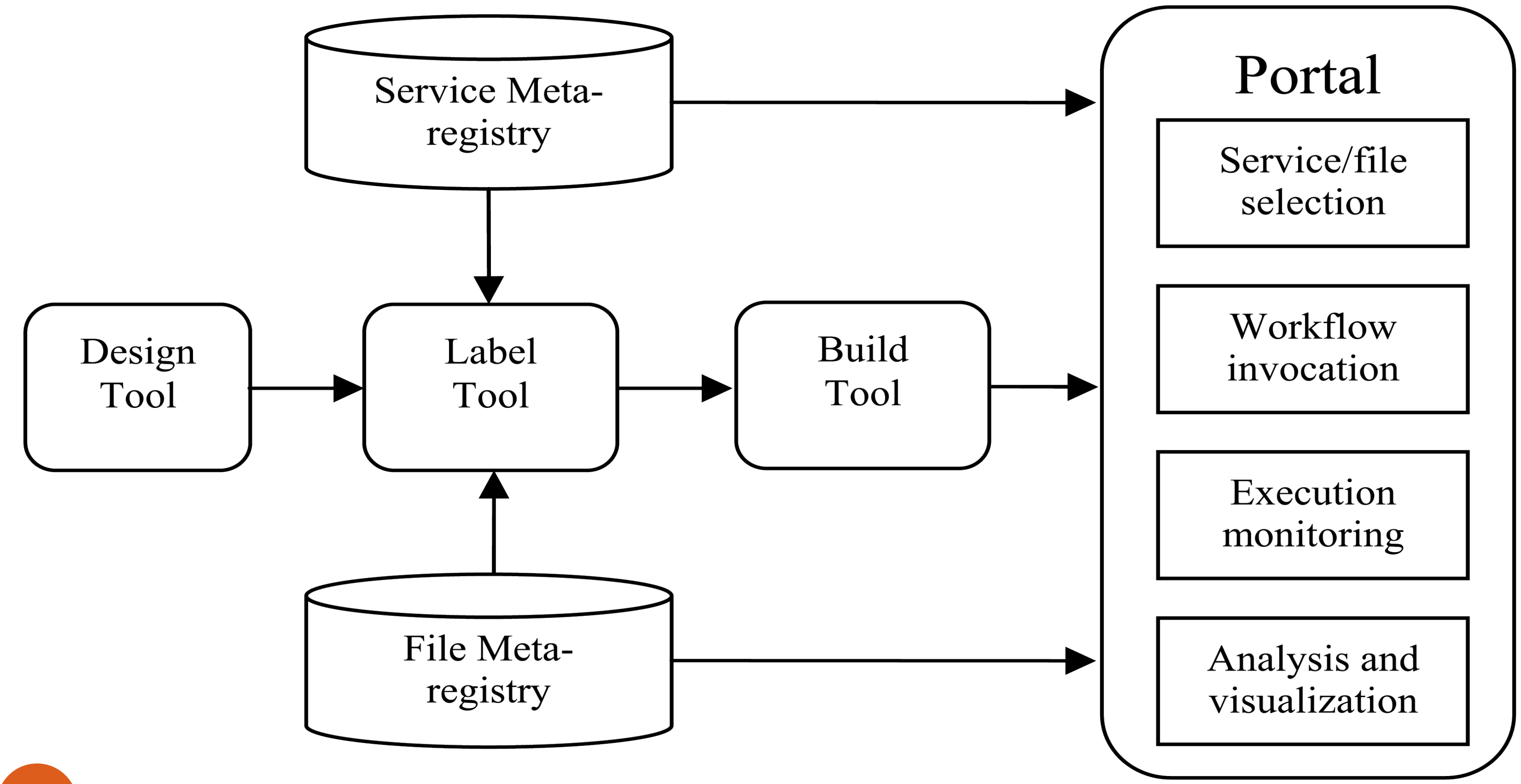


How do you insert Task 5 into this predefined workflow and how do you create a new portal for this new workflow?

A Problem and Solution

- Creating portals for different workflow structures is difficult for users who have no expertise in Web portal programming and Web services.
- We would like to automatically generate portal based on the XML workflow description.
  - Build a workflow structure (especially in a visual programming environment)
  - Label services and file types in the workflow
  - Generate a portal environment

The Components of the Portal Generation System



Design Tool

- Is a visual programming environment
- Draws the skeleton workflow
- Converts the skeleton workflow into an XML file that conforms to a specific XML schema.

Label tool

- Connects to web service and file repositories.
- Associates a file or service type with each node of the skeleton workflow.
- Creates an XML file describing the abstract workflow.

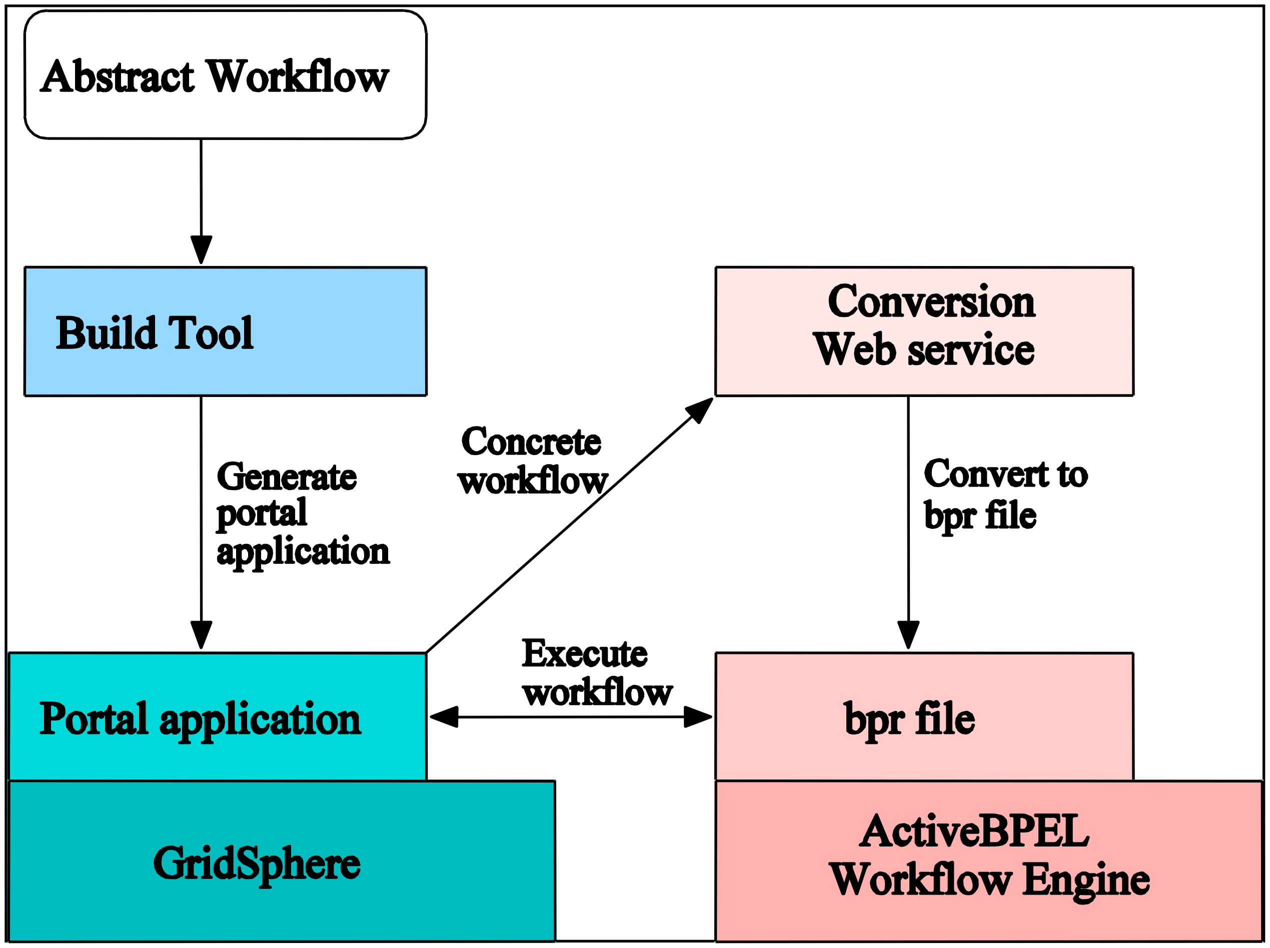
File and Service Types

- A set of files are said to have the same file type if their content conforms to the same specified template or schema.
- A set of services are said to have the same service type if they have the same typed interface and perform the same high-level computational task.
- In this context, saying that a set of services have the same service type is the same as saying that they are “semantically-equivalent services”.

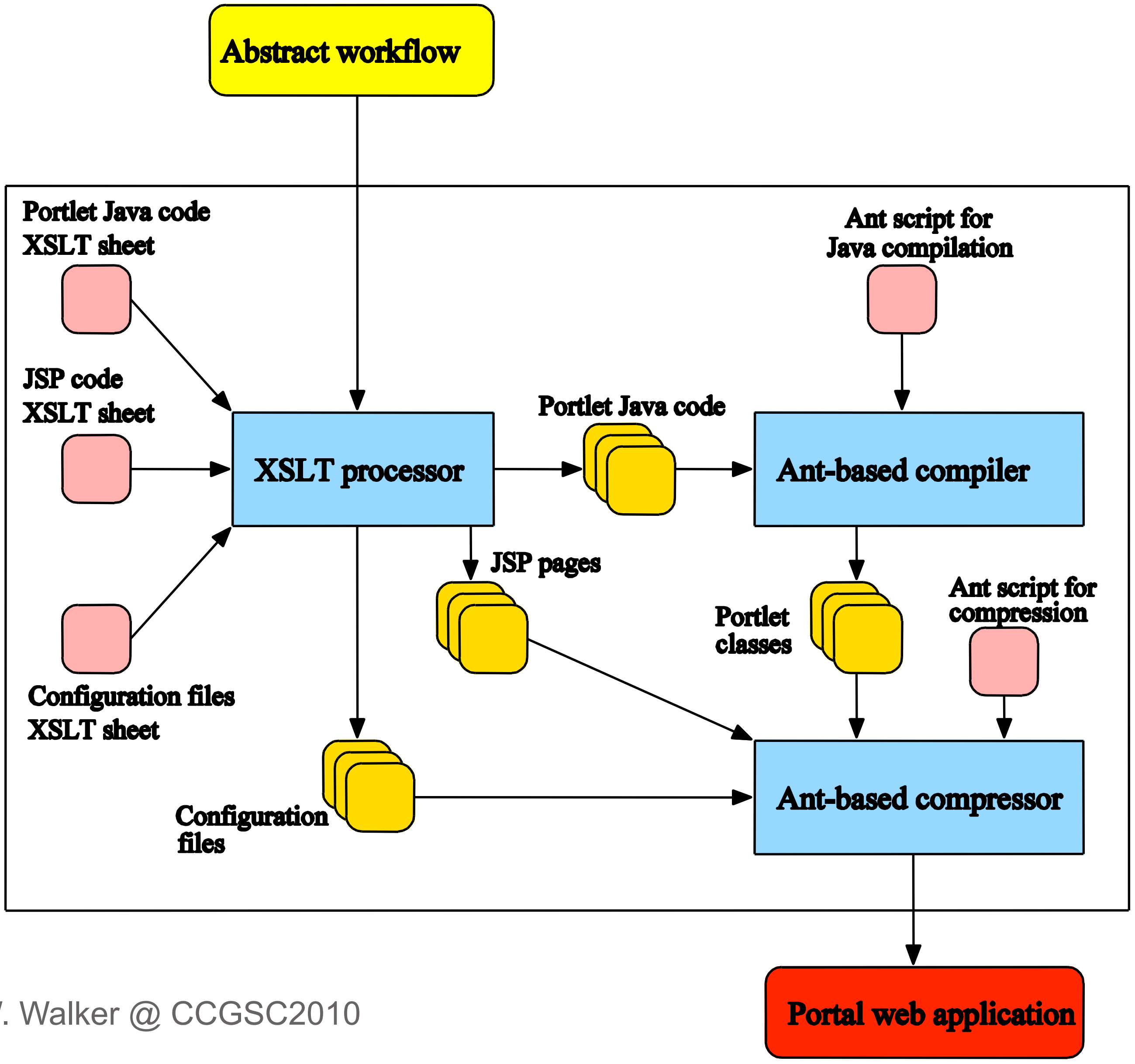
Build Tool

- Retrieves the details on the sets of web services and files registered in the file and service repositories, respectively.
- Creates a new portal corresponding to the abstract workflow description file created by the Label Tool.
- The Build Tool uses XSLT technology to convert an abstract workflow into a portal WAR file.

Prototype System



Structure of the Build Tool



Portal Generated by the Build Tool

- The portal is used to generate a service-oriented application by selecting different web service instances and creating an activeBPEL concrete workflow description file.
- The activeBPEL file is then deployed on a workflow execution engine via the network.
- The portal also provides users with a portlet to execute the workflow and monitor the result.

Future work

- Multiple portal framework support (JetSpeed, WebSphere, etc)
- Multiple third-party workflow engine support
- Workflow execution optimization
  - Asynchronous service invocation on dual HTTP channels
  - More checkpointing support.

Publishing Research Results and Data

- Living documents are a novel way of electronically publishing research results. Readers can replay simulations, and experiment with changing input parameters.
- Embed links to web interface in paper (in pdf for example).
- Specify input data in web interface.
- Receive back plot of results in browser or via email.

Living Documents

- Input data form invokes Java servlet.
- Java servlet acts as resource broker for accessing distributed resources (clusters, grid, cloud, Condor).
- Can exploit coarse-grain parallelism.

Provenance Support

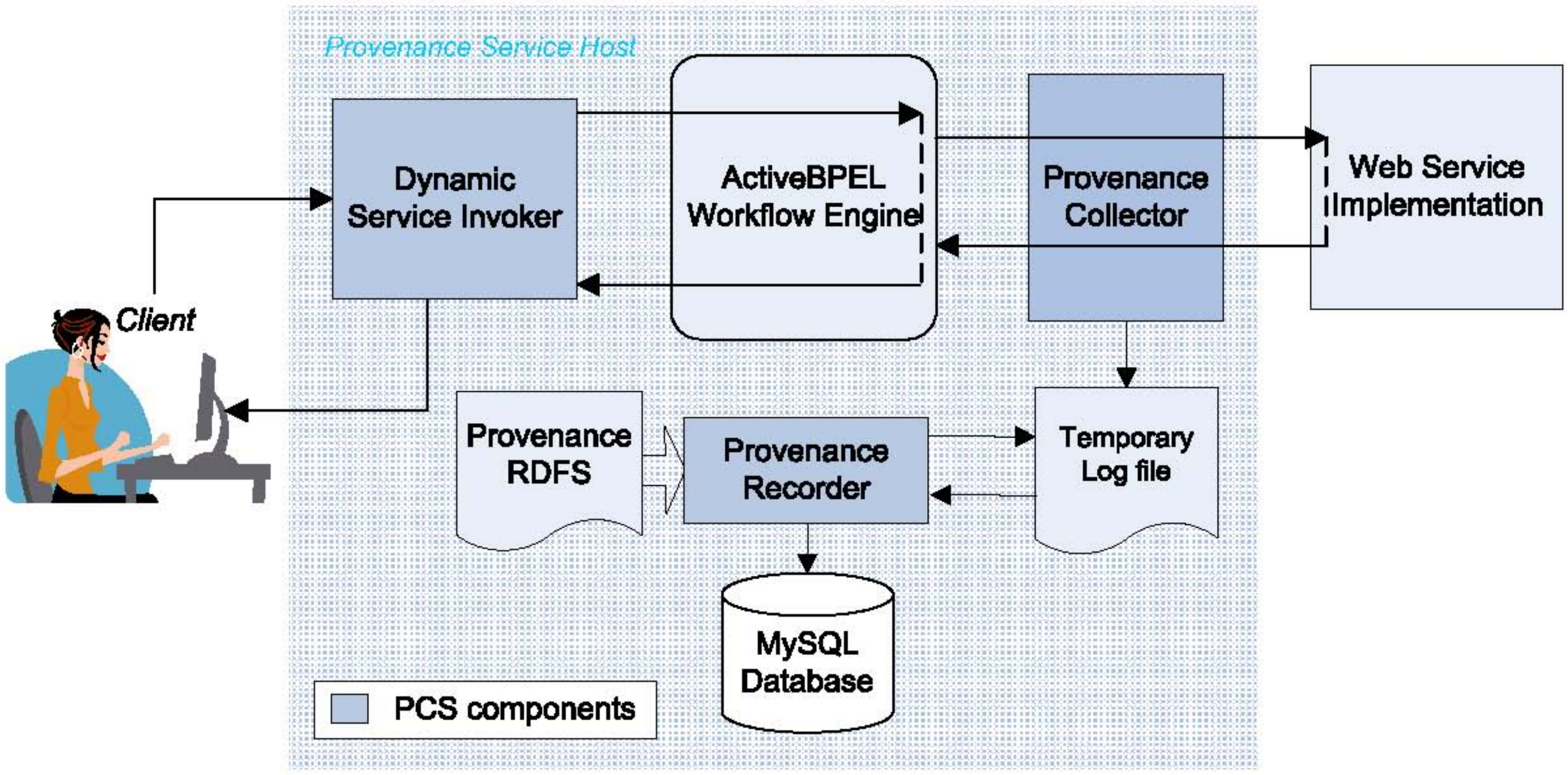
- When a workflow executes in a service-based environment we want to:
  1. Document data provenance for the data created by the execution.
  2. Use the provenance documentation as a recipe to re-execute the workflow.
- This is done through a Provenance Service.

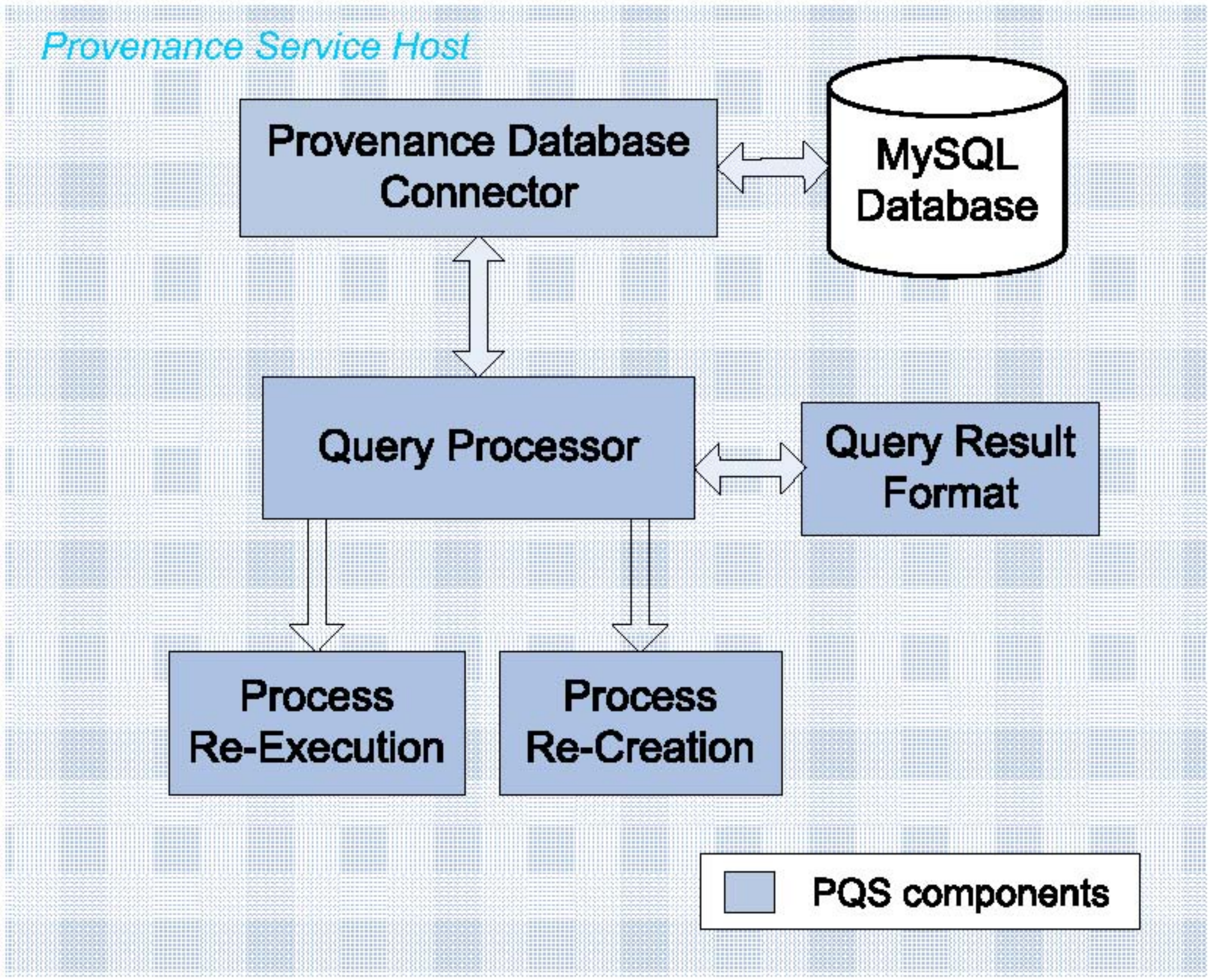
Provenance Service

- A client wishing to capture the provenance of his/her process execution submits the file describing the Web service composition to the Provenance Service.
- The Provenance Service is responsible for:
  1. Interacting with the workflow engine for the execution of the composite workflow.
  2. Capturing the provenance of the process execution.
  3. Recording the captured provenance of the process.
- The Provenance Service returns the final result data of the execution and informs the client of completion.

Use of Provenance Data

- The client is allowed to browse and query the recorded provenance of the processes that s/he previously executed.
- The client is able to validate previously-run processes, the incorporated web services, and the returned output data through re-execution of the process via its provenance.
- The client can also change the input parameters of the process during re-execution to perform "what-if" analyses.


Provenance Service Host
Dynamic Service Invoker
ActiveBPEL Workflow Engine
Provenance Collector
Web Service Implementation
Client
Provenance RDFS
Provenance Recorder
Temporary Log file
MySQL Database
PCS components


Provenance Service Host
Provenance Database Connector
MySQL Database
Query Processor
Query Result Format
Process Re-Execution
Process Re-Creation
PQS components