



IST-2001-32133

GridLab - A Grid Application Toolkit and Testbed

## D 3.3 Triana WorkFlow Specification

---

Author(s):	Ian Taylor, Shalil Majithia, Matthew Shields, Ian Wang
Document Filename:	GridLab-3-D3_3-0001-WorkFlowSpec
Work package:	WP3 Work-Flow Application Toolkit (TGAT)
Partner(s):	University of Wales, Cardiff
Lead Partner:	University of Wales, Cardiff
Config ID:	GridLab-3-D3_3-0001-1.0-Draft_A
Document classification:	INTERNAL

---

**Abstract:** This document sets out the specification of the workflow system to be used within Triana.



## Contents

<b>1</b>	<b>Overview</b>	<b>2</b>
<b>2</b>	<b>Main Issues</b>	<b>3</b>
2.1	Triana GUI . . . . .	5
2.2	TaskGraph Representation . . . . .	6
2.3	Triana Service . . . . .	6
<b>3</b>	<b>Other Issues</b>	<b>7</b>
3.1	Control Constructs . . . . .	7
3.1.1	Parallel Processing . . . . .	7
3.1.2	Conditional Processing . . . . .	8
3.1.3	Iteration . . . . .	9
<b>4</b>	<b>Appendix A: Definitions</b>	<b>11</b>
<b>5</b>	<b>Appendix B: TaskGraph Schema</b>	<b>12</b>

## 1 Overview

This section outlines the concept of workflows. A workflow management system (WFMS) is a generic software tool which allows for the definition, execution, registration and control of workflows (cf. [1]). The Workflow Management Coalition (WfMC) defines a workflow management system as follows:

A system that completely defines, manages, and executes workflows through the execution of software whose order of execution is driven by a computer representation of the workflow logic.

A workflow schema is the actual topology of a workflow, that is, the sequence of tasks which must be performed in order to complete a job. A workflow schema does not need to be a linear sequence-it can have decision points at which a job takes one of a number of possible paths. More interestingly, it can have points at which a job can split into multiple concurrent sub-jobs, and other points at which multiple sub-jobs can join to reconstitute a job. Within Triana, workflows are experiment-based, i.e. every piece of work is executed for a specific experiment. Experiments are generated by an external customer (e.g. a scientist using Triana). The goal of workflow management is to handle experiments as efficiently and effectively as possible. Experiments are handled by executing tasks in a specific order. The workflow process definition specifies which tasks need to be executed and in what order. A task which needs to be executed for a specific experiment is called a task. A task which is being executed by a specific resource is called a Runnable Task.

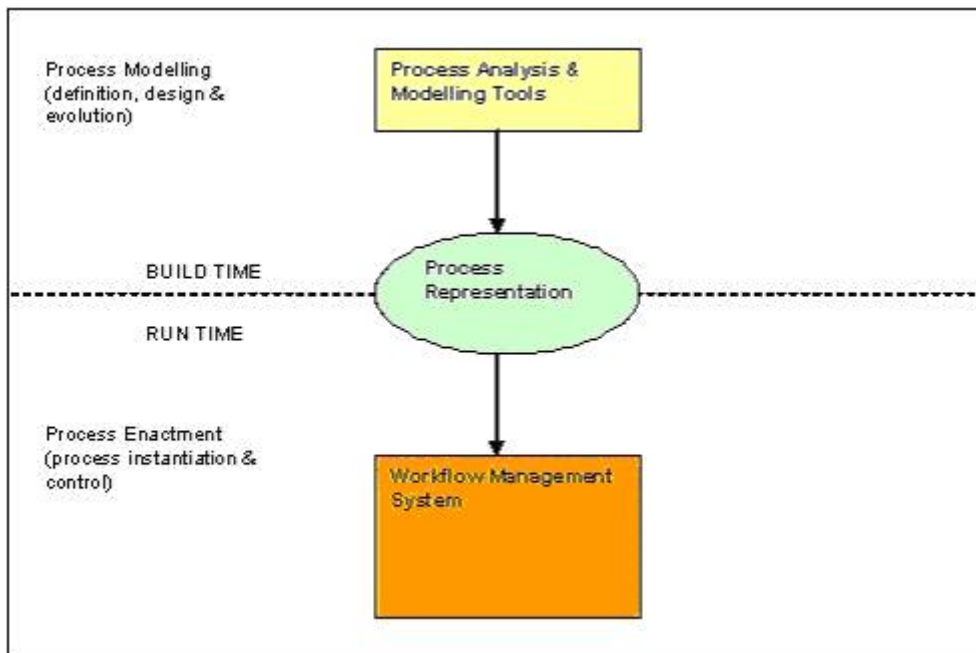


Figure 1: WfMC reference Model

Figure 1 shows an overview of the WfMC reference model. The reference model describes the major components and interfaces within a workflow architecture. The core of any workflow

system is the workflow management service. The workflow management service provides the run-time environment which takes care of the control and execution of the workflow. The process definition tools are used to specify and analyze workflow process definitions and/or resource classifications. These tools are used at design time. Most workflow management systems provide three process definition tools:

- a tool with a graphical interface to define workflow processes,
- a tool to specify resource classes (organizational model), and
- a simulation tool to analyze a specified workflow.

The end-user communicates with the workflow system via the workflow client applications. The reference model also incorporate administration and monitoring tools which are used to monitor and control the workflow. These tools are used to register the progress of cases and to detect bottlenecks. Also, these tools are used to set parameters and handle abnormalities.

The role of a WFMS is to provide the services required for automating workflow processes. WFMS systems can be characterized by the following functional components:

- modeling and representation of workflow processes and their constituent activities;
- selection and instantiation of processes for activation in response to user requests or key events;
- scheduling and tasking of activities;
- monitoring and adaptation of execution processes.

In this section, we looked at the meaning of a workflow and evaluated the WfMC reference model. In the next section, we look at how this model can be applied to Triana.

## 2 Main Issues

This section provides a brief overview of Triana architecture and a detailed look at how workflow is implemented within Triana. Triana is a two layered application consisting of Triana GUI and Triana Service. Triana GUI provides an easy to use graphical interface to constructing a network. Alternatively, a command line interface is also available for this purpose. Once the network is created, an XML based TaskGraph is generated. This TaskGraph is executed by the Triana controller through a gateway Triana Service, see Figures 2 and 3.

Specifically, a Triana Service is comprised of three components: a client, a server and a command process server. In a typical network, only one command process server will be active. The client component of the Triana Service in contact with the Triana Controller then pipes data and program to the server components of other Triana Services. These Triana services execute the tasks allocated and transfer data as prescribed by the Triana controller.

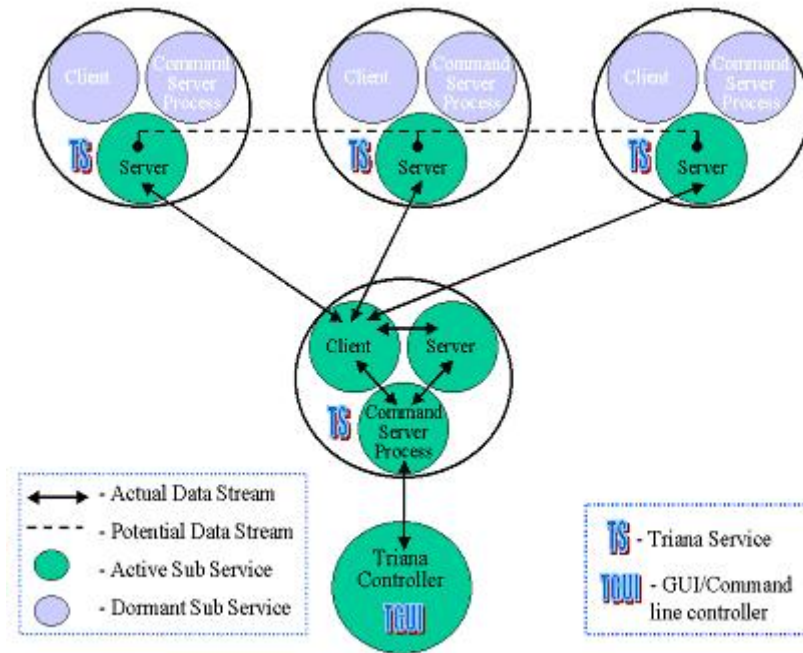


Figure 2: A Schematic representation of Triana

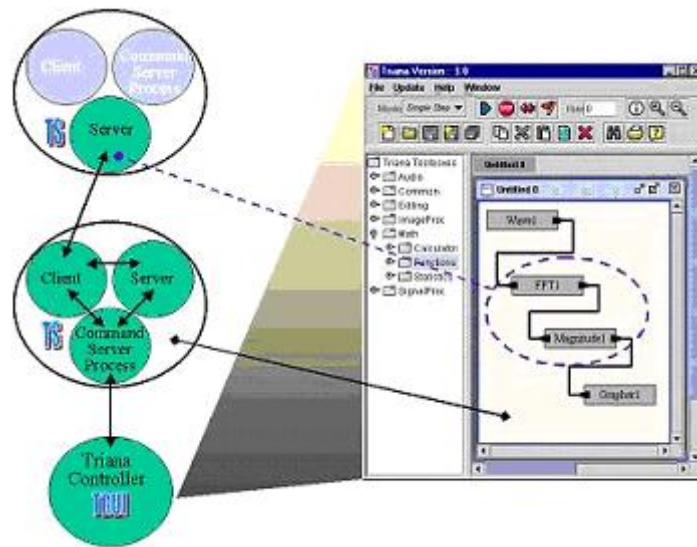


Figure 3: An Actual Implementation of Triana

Accordingly, the architecture of a grid aware Triana is as shown in Figure 4.

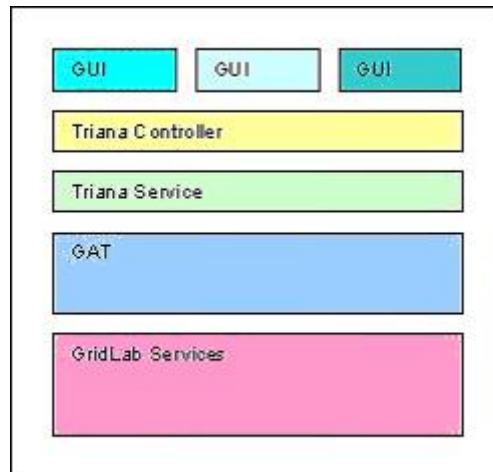


Figure 4: Triana Grid Architecture

Based on this architecture, three main issues can be identified:

- Triana GUI - a process modeling tool which provides the user with an easy to use drag and drop graphical interface to construct networks.
- TaskGraph Representation - A definition of the process comprising a number of discrete steps and expressed in a textual form. In the case of Triana, this is represented in XML WSFL-like format.
- Process execution - similar to a workflow engine, a gateway Triana Service executes the TaskGraph, making calls to the GAT as required.

These three issues are now considered in further detail.

## 2.1 Triana GUI

The Triana GUI provides the user with a means to construct a network by a simple drag and drop action, connect and group tasks, set and change parameters etc. At least three types of GUI are planned to be implemented: a standard version which will run on machines, a light version to run on resource constrained devices and a web browser view to report progress/change parameters etc. The Triana GUI also allows the user to specify the machine on which to run the entire/partial TaskGraph. Figure 5 provides a snapshot the Triana GUI.

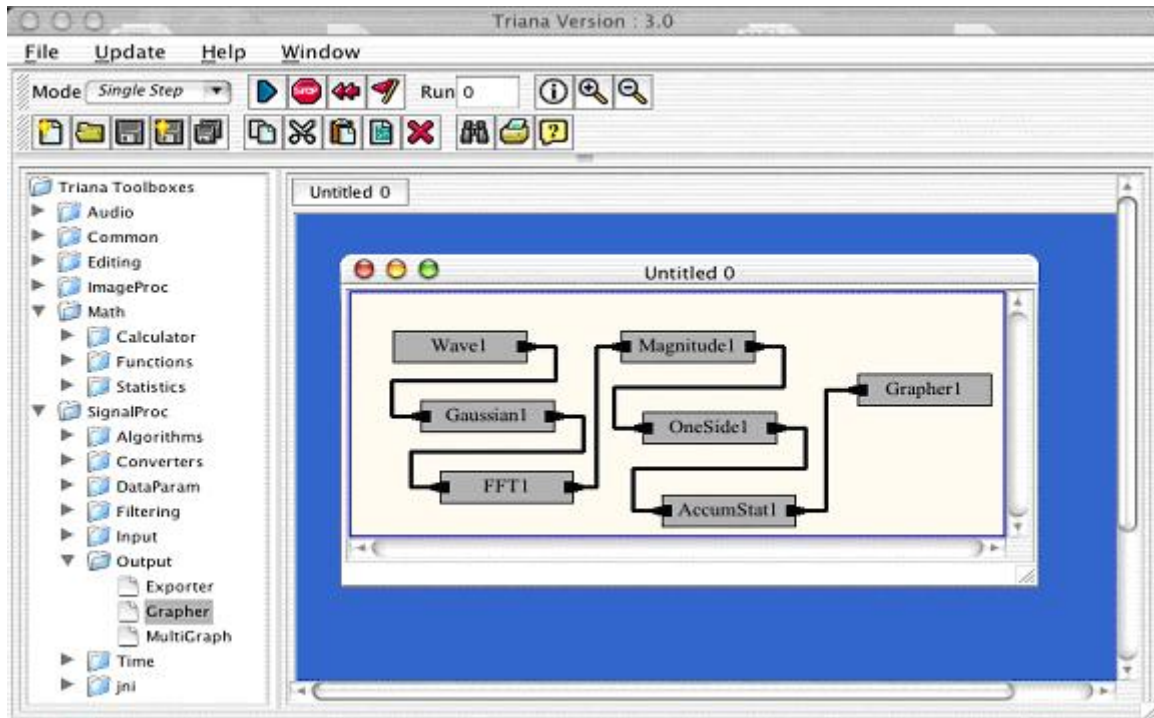


Figure 5: Snapshot of Triana GUI

## 2.2 TaskGraph Representation

In Triana, a workflow is represented by using an XML based WSFL like representation format. The Web Services Flow Language (WSFL) is an XML based language for composing and choreographing web services flow. WSFL is used to produce an XML based representation of a process. This representation is then fed into a middleware application designed to invoke and manage the process. WSFL uses WSDL for the description of service interfaces and their protocol bindings. WSFL also relies on an envisioned "endpoint description language" to describe non-operational characteristics of service endpoints, such as quality-of-service properties (Web Services Endpoint Language). As this write-up does not cover OGSA/Grid web services, no further reference is made to WSDL. This has the implication that WSFL as it stands cannot be used to specify the workflow until such time that WSDL is incorporated within the Triana architecture.

Consequently, a WSFL-like XML representation is used to represent the taskgraph. The TaskGraph has three types of elements: Tasks, Control Links and Data Links. A task represents an operation. Control Links define the sequence of tasks in the model. Data Links describe the flow of data between tasks.

A complete schema is shown in Appendix B.

## 2.3 Triana Service

The Triana Service is responsible for validating the incoming TaskGraph and executing it. Additionally, the Triana Service distributes partial TaskGraphs to other Triana Services running on remote servers through the GAT. Triana Service is responsible for:

- Interpretation of the process definition;

- Control of task instances;
- Navigation between Tasks (sequential, parallel, Conditional etc)
- Maintenance of workflow control data;
- Maintaining audit and history details.

### 3 Other Issues

This section covers other workflow related issues, specifically how control is routed through the network by the use of special units.

#### 3.1 Control Constructs

There are three control constructs which need to be represented within Triana:

1. Parallel Processing,
2. Conditional Processing and
3. Iteration.

##### 3.1.1 Parallel Processing

For example, two tasks B and C need to be executed but the order of execution is arbitrary. To model such a parallel routing, two building blocks are used:

1. the AND-split and
2. the AND-join

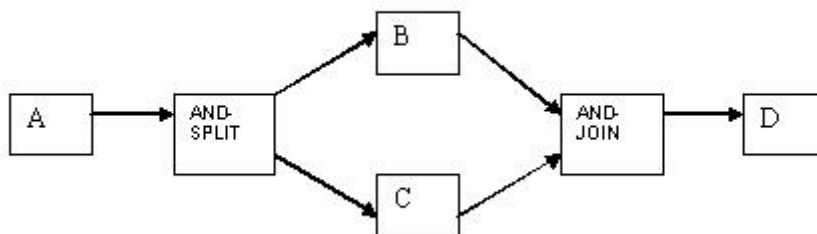


Figure 6: AND Construct

Figure 6 shows that both building blocks can be modelled by ordinary units/tools. The execution of AND-split enables both task B and task C. AND-join is enabled after execution of both B and C and is used to synchronize two subflows. As a result, task B and task C are executed in parallel. Parallel Processing is currently handled in Triana by allowing the algorithm programmer to specify the number of input/output nodes in a unit.

### 3.1.2 Conditional Processing

Conditional Processing is used to allow for a routing which may vary between cases. In this way, the routing of a case may depend on the workflow attributes. To model a choice between two or more alternatives, two building blocks are used:

1. the IF-split and
2. the IF-join

An IF-split can be modelled by a unit with multiple outgoing cables, an IF-join is modelled by a unit with multiple ingoing cables. Figure 7 shows the situation where task A is followed by either task B or task C, i.e., a choice is made between B and C. The execution of one of these two tasks is followed by the execution of task D.

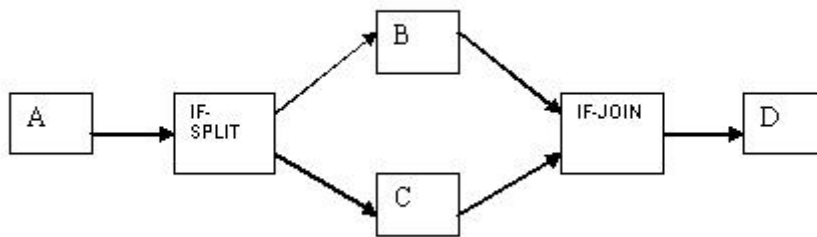


Figure 7: IF Construct

The choice between alternatives often depends on workflow attributes. If the choice is based on workflow attributes, it is a deterministic choice. One way to model a deterministic choice between B or C is shown in Figure 7. Task A passes the variable which is to be tested and the data to the IF-SPLIT unit. The user can specify the parameter to be tested and accordingly the data is routed to the appropriate unit.

Additionally, the SWITCH control construct is used to specify which of two inputs will be used by a visual network, depending on a conditional expression. Data will be obtained from the first input if a specified condition is TRUE; it will be obtained from the second path if the specified condition is FALSE.

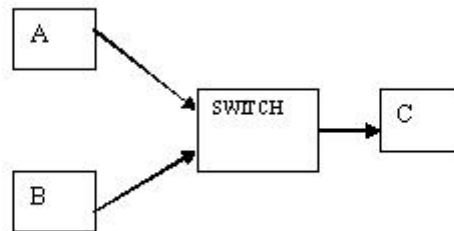


Figure 8: SWITCH Construct

### 3.1.3 Iteration

Looping constructs cause control flow of the visual program to iterate through a particular section of the network multiple times. Looping constructs depend on the value of variables to determine how many times the control flow will iterate on the specified section of the network. Two possible looping constructs are:

#### 1. Count Loop

The Count Loop causes the data flow to iterate through a particular section of network a specified number of times.

When data is available at the input connection(s) of the count loop unit, events will occur as follows:

- First, the counter will be initialized.
- Next, the current value of the counter will be compared to the final value specified; if the current value is less than the final value, the network contained in the count loop will be executed.
- After the units within the count loop are executed, the increment value specified will be added to the current value of the count variable.
- The new value of the count variable will again be compared to the specified final value; if the current value is still less than the final value, the network will be rescheduled. This process will be repeated until the current value of the count variable meets or exceeds the final value specified.
- Data flow will then progress downstream from the output connection(s) of the count loop unit to the rest of the network.

#### 2. While Loop

The While Loop causes the data flow to iterate through a particular section of network as long as a particular condition is met.

When data is available at the input connection(s) of the while loop unit, events will occur as follows

- the exit condition will be evaluated; if it evaluates as FALSE, the network contained in the while loop is executed.

- After the units in the while loop are executed, the conditional expression (the exit condition) will then be evaluated again; if the conditional expression is still FALSE, the network will be rescheduled; the process will be repeated until the conditional expression is evaluated as TRUE.
- Data flow will then progress downstream from the output connection(s) of the while loop unit to the rest of the network.

Within Triana, the user creates a Group by selecting the units to be looped over. The Group makes available all the parameters to the user so that the user can set up exit conditions. Figure 9 shows a graphical representation of a loop.

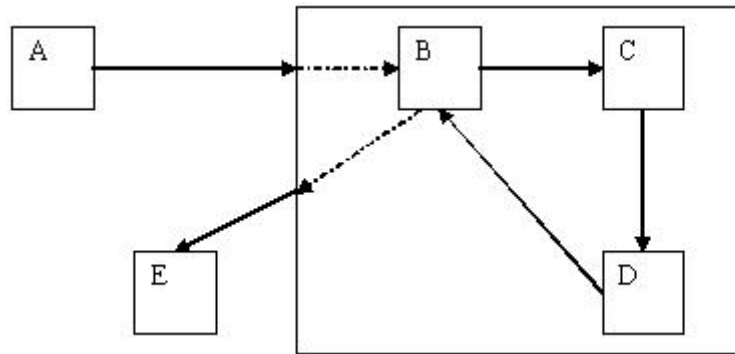


Figure 9: LOOP Construct

## 4 Appendix A: Definitions

*Workflow* - Sequencing of tasks which need to be performed in order to accomplish a specific goal.

*Task* - An object that does something, has inputs and outputs e.g Grapher, FFT etc

*GroupTask* - A recursive composition of tasks, i.e. a task which contains other tasks

*TaskGraph* - A workflow process definition - defines the tasks that need to be executed and the order in which they are executed.

## 5 Appendix B: TaskGraph Schema

```
<?xml version="1.0" encoding="UCS-2"?>
<!ELEMENT taskgraph (name, description, parameters, tasks, controllink, dataLink)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT parameters (param*)>
<!ELEMENT param (#PCDATA)>
<!ATTLIST param name CDATA #IMPLIED>
<!ATTLIST param value CDATA #IMPLIED>
<!ELEMENT tasks (task*)>
<!ELEMENT task (inportnum, outportnum, description, input, output, parameters)>
<!ATTLIST task toolname CDATA #IMPLIED>
<!ATTLIST task taskname CDATA #IMPLIED>
<!ATTLIST task taskid CDATA #IMPLIED>
<!ELEMENT inportnum (#PCDATA)>
<!ELEMENT outportnum (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT input (type)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT output (type)>
<!ELEMENT controllink (source, target)>
<!ELEMENT dataLink (source, target)>
<!ELEMENT source (#PCDATA)>
<!ELEMENT target (#PCDATA)>
<!ATTLIST source taskid CDATA #IMPLIED>
<!ATTLIST source node CDATA #IMPLIED>
<!ATTLIST target taskid CDATA #IMPLIED>
<!ATTLIST target node CDATA #IMPLIED>
```

## References

- [1] Lawrence, P, Workflow Handbook 1997, Workflow Management Coalition, Wiley and Sons, New York