

# An Applications Framework for Astronomical Data Processing and Analysis

Doug Tody<sup>1,2</sup>, Preben Grosbøl<sup>3</sup>,  
Bill Cotton<sup>1</sup>, Malcolm Currie<sup>5</sup>, Mike Fitzpatrick<sup>6</sup>, Luigi Paioro<sup>4</sup>, Christian Surace<sup>7</sup>

<sup>1</sup>National Radio Astronomy Observatory, <sup>2</sup>US NVO, <sup>3</sup>European Southern Observatory, <sup>4</sup>Italian National Institute for Astrophysics,  
<sup>5</sup>Rutherford Appleton Laboratory, <sup>6</sup>National Optical Astronomy Observatory, <sup>7</sup>Laboratoire d'Astrophysique de Marseille

## Introduction

The initial phase of planning for a Future Astronomical Software Environment has been completed, led by OPTICON in Europe and the National Virtual Observatory in the US. The second phase of the project, focusing on implementation of the **applications framework** is expected to go forward under a joint US and European team beginning in 2009. In this paper we provide an update on the system architecture and present a conceptual design for an actual system compliant with this architecture, identifying the major system elements and interfaces and describing their functionality.

The applications framework project is an attempt to provide a common desktop and server framework to bring observatory data processing for data from multiple observatories into a common environment, to improve functionality and interoperability as well as reduce costs, and help integrate observatory data processing with VO and distributed multiwavelength data analysis. A multi-level effort is planned, including development of standards for the key elements of the framework, implementation of reference implementations of the major system elements, and pilot projects by the groups involved to use the framework for real use cases.

The project described here is a joint effort of the OPTICON network in Europe and the NSF-funded NVO project in the US, with participation from groups carrying out related prototyping efforts (e.g., ESO, INAF, LAM, NOAO, NRAO).

## Use Cases

The primary use cases we identified are the following:

### Desktop data processing and analysis

The framework is used to provide a desktop environment for processing and analysis of astronomical data. A common framework is desirable to provide a more uniform user experience and to promote interoperability of software from multiple observatories. Data interaction may include hands-on reprocessing (e.g., by PIs) of instrumental data, and integration with VO for access to remote data for analysis. The actual processing may be performed either locally or remotely, driven from the desktop.

### Pipeline processing

For PI observing programs on modern telescopes it is often desirable for the astronomer to be able to manually reprocess and tweak their data, even though a standard data reduction pipeline may already be provided. The same software should be used in both cases. Execution may take place either remotely, or locally on the user system, hence it is necessary to be able to export the pipeline software to the user. For this to work well the desktop system and the pipeline system should be based on the same software.

### Data Processing for VO Services

This includes both production of virtual data products for VO services as well as implementation of data analysis services. Both are similar computationally to conventional astronomical data processing, hence it is desirable to be able to re-use the same software for both cases. The applications framework described here deals only with the computational problem; existing VO standards would be used to publish data to the VO, provide a Web interface for constructing Grid workflows, and so forth.

A common computational framework is desirable to integrate software for these different use cases and enable software sharing.

## Applications Architecture

An applications architecture defines the form of applications and how they are constructed. Applications in our architecture have the following characteristics:

### Application Package

Large systems may contain hundreds of applications, often from multiple sources or "package vendors". Often multiple versions of applications must be supported. To make this problem manageable, related applications are grouped into **applications packages** which can be separately managed.

### Application

An **application** is some top level program which is directly usable to solve some specific problem. The applications framework does not define the form of a top level application, rather it defines the facilities available for constructing such applications. In our architecture applications are often written in a high level language calling **components** (tasks) for much of the functionality.

### Task or Tool

A **task** is a simple component, such as a data processing module, which can be called from an application to perform some function (in the limiting case the task itself may be the application). A **tool** is a more general, stateful type of component.

In a typical scenario a top level application would be written in a high level language such as Python (for science apps) or Java (for VO or Web apps). The application would call libraries of tasks to perform computational "tasks" (hence the name). Numerically intensive tasks would often be written in compiled languages, or possibly hybrids such as Python+C. The framework is responsible for communications, and distributed and scalable execution of the application.

## Framework Elements

For an applications framework to support applications development fully one needs both the runtime framework as well as class libraries for the language of choice, supporting the desired applications domain (O/IR, radio, X-ray, VO, etc.), and finally a build system. While the full scope of our project includes all of these, our concern here is with the runtime execution framework. The major elements of the runtime framework are as follows:

### Packaging

The **packaging** mechanism defines the composition of an applications package, including metadata describing the package including all tasks and their parameters, as well as executable versions of all modules and any other runtime files required for execution.

### Component Container

All **components** (tasks and tools) run within a standard **container**. Containers may be in the same process as the application, or may be distributed. Tasks may be serial or parallel, and may be written in any supported language.

### Parameter Mechanism

The inputs and outputs (in terms of framework execution) for both tasks and tools are defined using a standard **parameter** mechanism. Parameters are grouped into **parameter sets**, which may be manipulated as objects and which may be persistent.

### Messaging

Applications and components may communicate at runtime via **messaging**. Messaging is also basis for the execution framework itself, and is required to be able to manage task execution. Open-protocol **messaging** in the framework will probably use SAMP (2.0).

### Package Manager

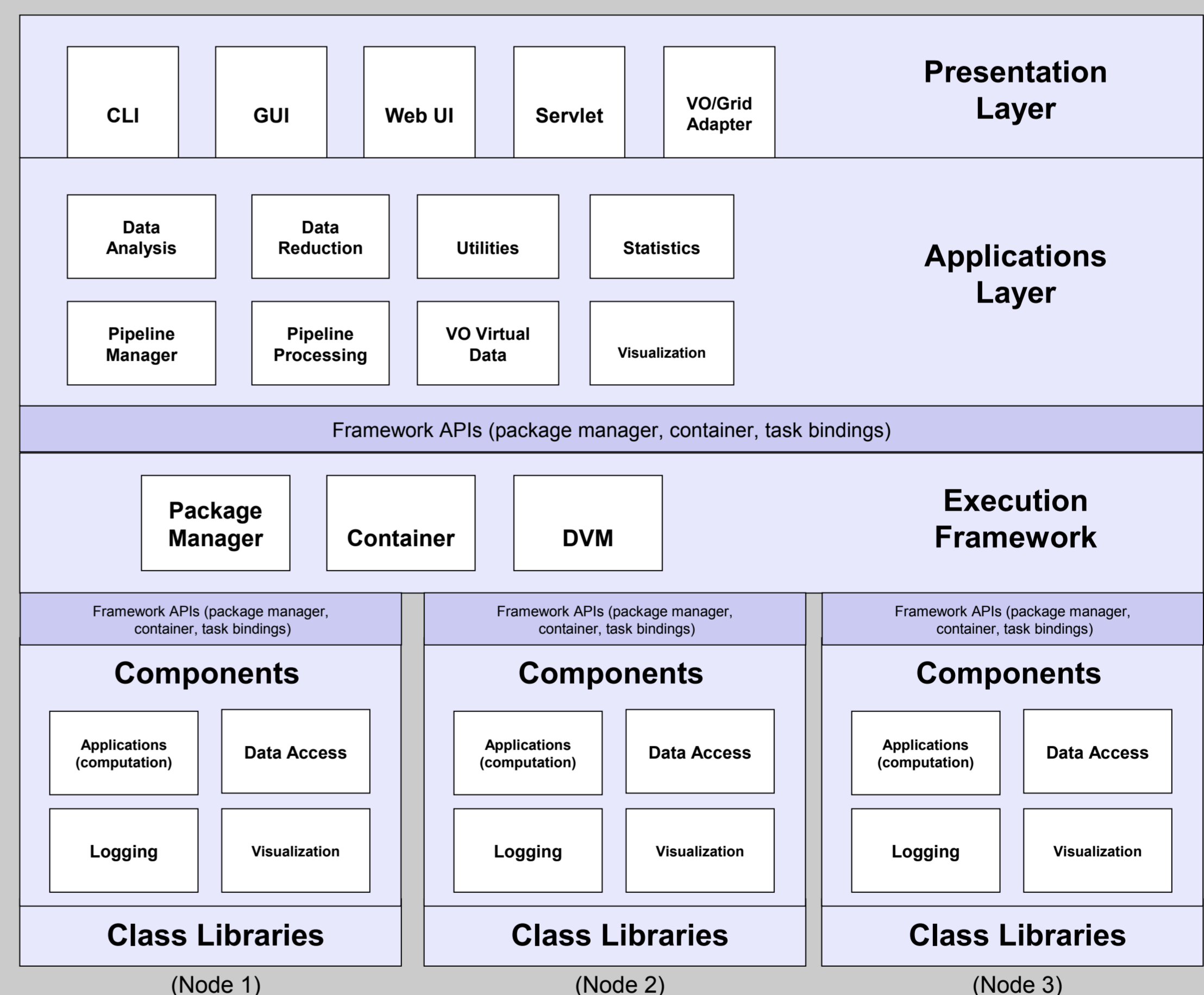
Management of all package metadata including parameter sets, as well as execution of tasks, is provided by the **package manager**. The package manager also provides the primary client interface to the framework and to the packages it manages.

### Distributed Virtual Machine (DVM)

In the most fully-featured, scalable version of the framework, distributed execution is provided by the DVM. In simpler single-computer configurations the DVM largely disappears, but some distributed execution is still possible.

### Framework Services

The runtime framework will also provide some standard services, e.g., for logging and for mediating access to shared data objects.



## Implementation Plans

Implementation of the applications framework described here is expected to go forward in 2009 and will take several years. While this will be an open standards effort in which anyone is welcome to participate, resources for the core effort as currently defined will come from the US VAO and OPTICON. In addition, partners, mainly drawn from the major public observatories within the US and Europe, will participate by using the framework to implement project-specific use cases, feeding the results of these efforts back into the standards development process. Standards development will be coordinated with the international VO projects and the IVOA, however the scope of the project is broader than VO as a major goal is to directly support observatory data processing.

The main products of the core effort will include specifications and reference implementations of the key elements of the the applications framework. This would make it possible to provide access to important legacy applications (e.g. from AIPS, CASA, IRAF/PyRAF, MIDAS, Starlink) with the help of the respective groups. Complete use-case applications based upon the framework will require additional resources from the partner projects!

