An integrated Data Mining framework for massive datasets

M. Brescia[1,2] Omar Laurino[1] G. Longo[1,2]

[1]Department of Physical Sciences, Napoli, Italy
[2]INAF, Napoli, Italy

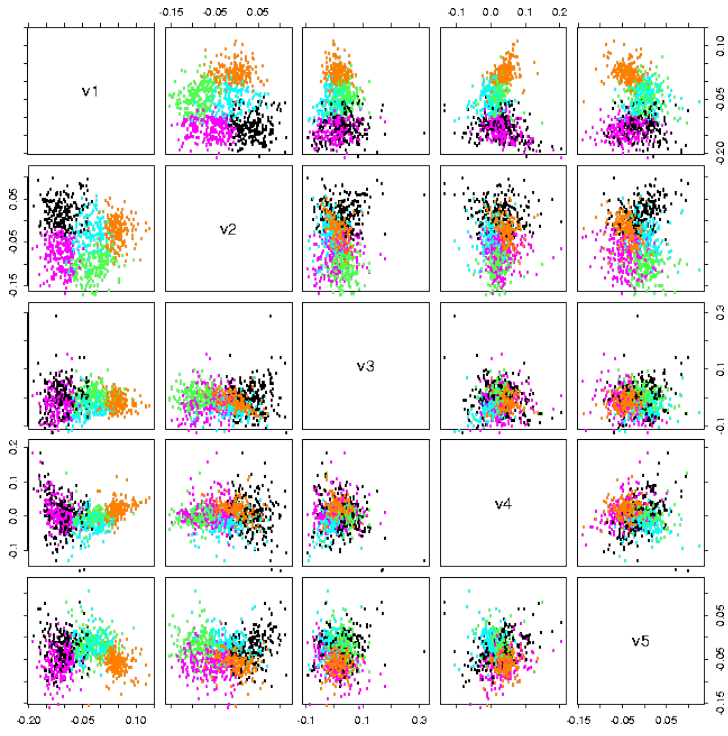
Caltech January 12, 2009

Outline

1. **Introduction**
   - Data Mining
   - What we've done so far
   - Goals and Requirements
2. **VONeural 2.0 – DAME**
   - Using VONeural2.0/DAME — The Front End
   - General Architecture
   - The Framework
3. **Extending VONeural 2.0 — DAME**
   - Developing Methods and Science Cases
   - Developing Drivers
   - The Deployment Environment Abstraction Layer
4. **Future Work and Developments**
   - Where to go now?

Why should I care about Data Mining

- Data is “growing”:
  - in volume;
  - in dimensionality;
  - in quality.
- Data can’t always be *explored* or *understood* by means of analytical tools (in reasonable time).
- Data may contain “unknown” or “rare” objects (outliers).



Knowledge Discovery in Databases $\Longrightarrow$ Data Mining

Data Mining Applications

In Astronomy

- Star/Galaxy Classification.
- Photometric Redshift Estimation.
- Candidate QSOs extraction.
- Real Time transient classification.
- Image segmentation.
- ...and we are still looking for new challenges...

In other Fields

- Mass Scale Medical Screenings.
- Time Series Forecasting.
- Many many others...

VONeural 1.0 (1/2)

Mission

A toolbox for astronomical Data Mining.

Deployment

Astrogrid CEC executables.

Applications

- MLP
- SVM
- MLP2GRID
- SVM2GRID
- VONeural GRID Broker

VONeural 1.0 (2/2)

Limitations

- Too much dependent on astrogrid standards (not much "cloud", too much client-sided)
- Little flexibility (many ad hoc, non general solutions).
- "sort of... chaotic" growth.
- General purpose platform... not datamining oriented.
- No display of intermediate results.
- Problematic interface between middlewares.
- Oriented to world astronomical community only.

One year ago we decided to start to design a new data mining infrastructure...

Who is this VONeural2.0/DAME person anyway?

A joint venture between Napoli and Pasadena.
An integrated framework for datamining on massive datasets.

Target

- Users
- Data Miners
- Developers

Goals (1/3)

User Friendliness

User = MSS (Mean Square Scientist)

Developer Friendliness

Developer = Data Miner || Computer Scientist || Power Scientist

Extendability

- DM Models
- (G)UIs
- Deployment Environments

Goals (2/3)

To gather as many buzzwords as possible:

- Web 2.0
- Cloud Computing
- Grid Computing
- Service Oriented Architecture
- Life, the Universe and Everything *else*

Goals(3/3)

- Ensure mantainability.
- To ease the process of adding new methods.
- To ease the inclusion of other tools (visualization, statistics, etc...).
- Oriented to scientific community as a whole.
- Ensure Scalability (parallel computing, huge storage, and so forth).

Given the international nature of the project, the accurate definition of standards and interfaces is necessary to let all the involved partners (Caltech, India, UNINA) to contribute.

The MSS perspective

User interaction is (supposed to be) pretty straightforward.

- Signs up (Register and Confirm Registration).
- Manages files in the Virtual FileStore, in different Working Sessions.
- Browses and launches experiments.
- Monitors Experiments status and intermediate results.
- Retrieves results.
- Updates his Facebook profile.

Front End Specifications (1/2)

Different Front End implementations

- Web Application.
- Desktop Application.
- CLI package.
- Browser AddOn.
- Mobile/Embedded device application.

Communication Protocols

- REST $\longrightarrow$ HTTP(s) methods.
- Contextual VOTable specifications.
- Basic privacy control.
- Intermediate/final output retrieval

Front End Specifications (2/2)

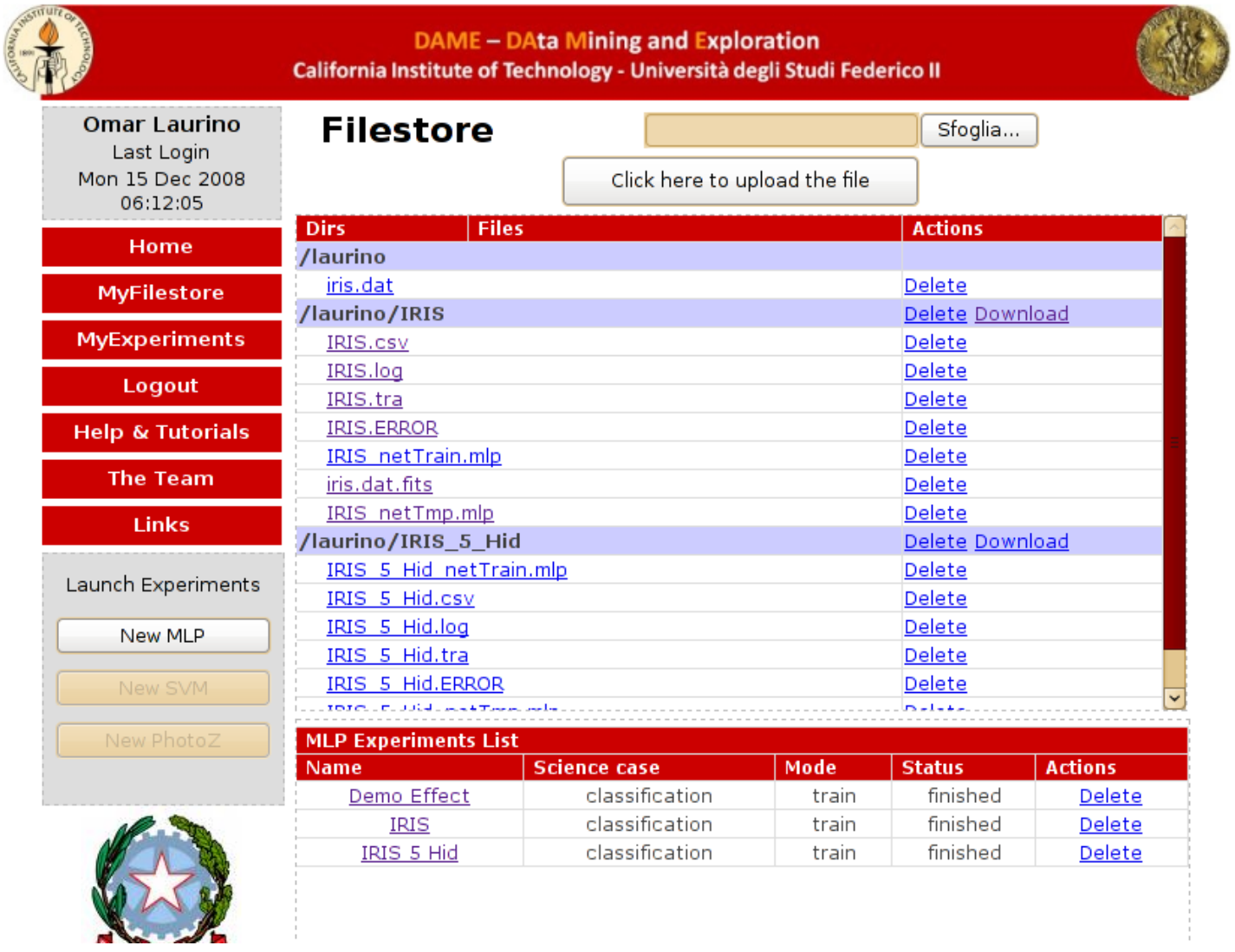
General Requirements

- RESTful WS client.
- Virtual FileStore.
- Methods Browser — Experiment launching pad
- Session Manager, Intermediate output retrieval and display
- Column selection/tagging
- Users Sign up/in interface.
- Method's input form rendering.

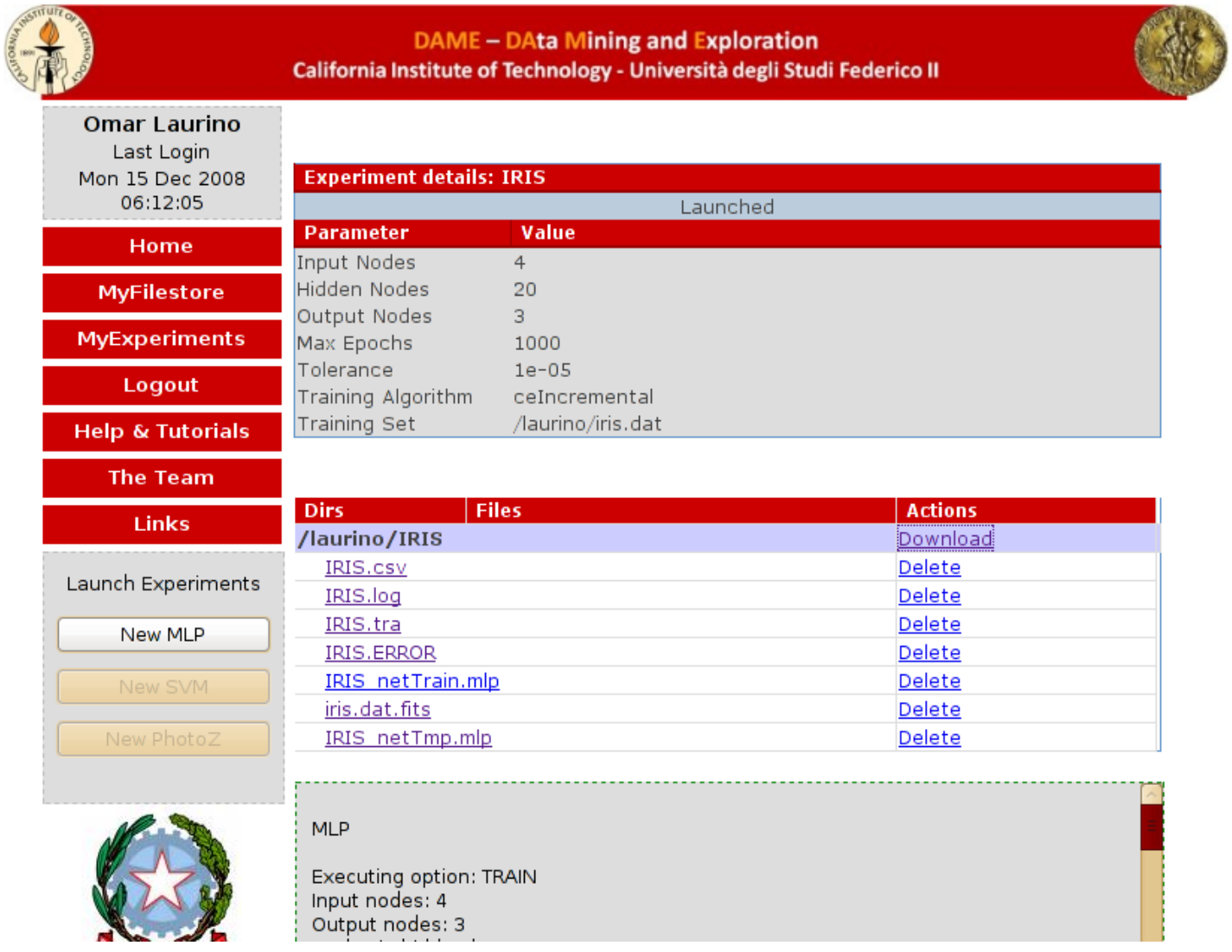
More specific requirements

- Advanced/Interactive visualization.
- Plastic/SAMP.

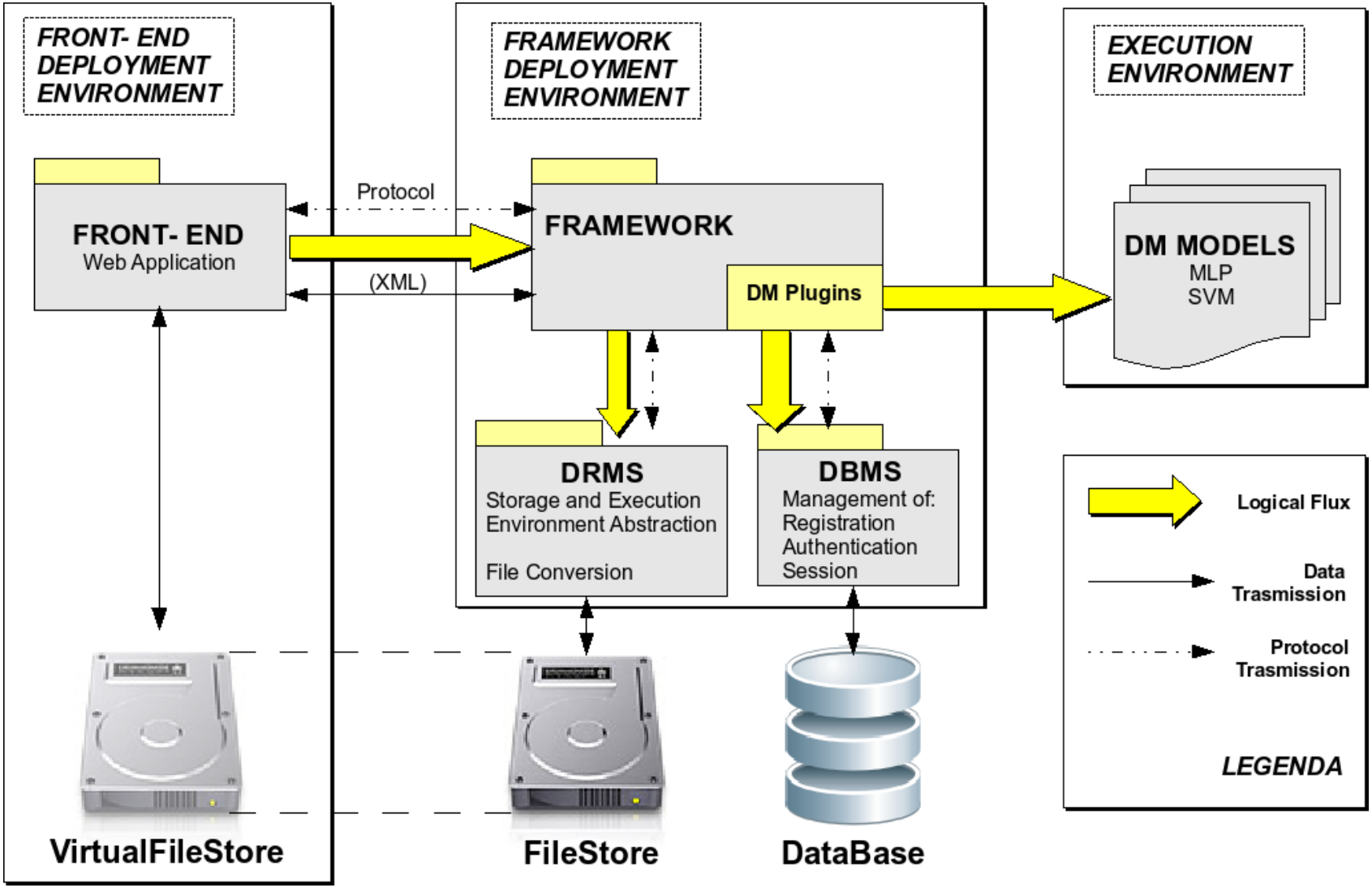
The first prototype 1/2



The first prototype 2/2



Component Diagram



The Framework Web Service

RESTful WebService

- Resources are “contextual” VOTables.
- File I/O: the (Virtual) FileStore.
- Experiments configuration and launch.
- Users Management.
  - Registration
  - Authentication
  - Authorization
- Working Session Management.
- Intermediate results retrieval.
- The WS just triggers atomic operations for the specific subsystems.
- Persistance is provided by a DBMS (the Registry).

Data Mining Plugins

Scientific Use Cases are implemented as Plugins, so they can be safely developed outside the infrastructure:

- Abstract Class to be implemented by means of a full SDK.
- Each plugin is registered by the FW admin in order to be exposed.
- During registration, a plugin description document is stored.
- The plugin is configured on the FW machine, than it can be serialized and sent to the execution environment.
- Communication with the FW requires a socket and the plugin is the client (much general).
- Plugins can be run with different “scheduling” priorities.

The Development Stack

Bottom-up development stack

- DM Plugin
- General class hierarchy (visualization, stats, etc...)
- DMM class hierarchy
- DM library wrappers (e.g. JNI)
- DM library (e.g. libfann, libsvm, etc...)
- Low level library (e.g. gsl, blas, etc...)

A Stat/DM Ontology

An integrated, complete and consistent DM framework can't lack statistical methods and a visualization library. Interactive visualization can happen on the Front End component only.

The science Developer perspective

A Developer wanting to extend the framework can:

DM Models Development

- Download our DM Models library.
- Add new low level/DM shared libraries.
- Add new wrappers.
- Extend the DM class hierarchy.

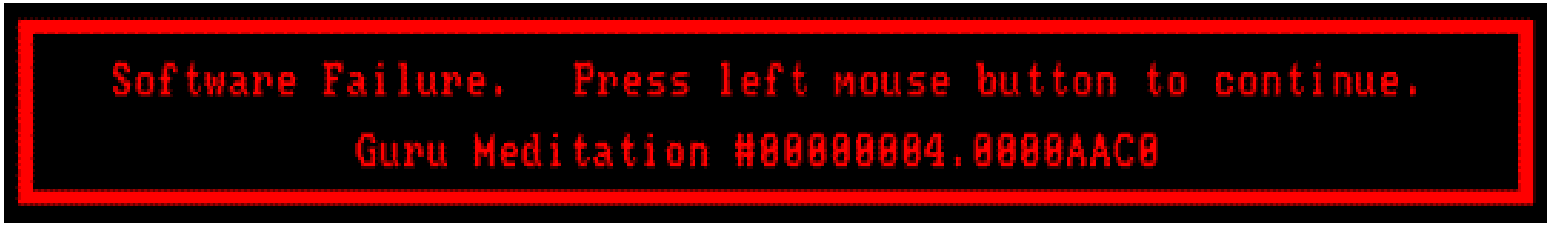
Plugin Development

- Download our SDK.
- Implement the DMPlugin abstract class.
- Test it.
- Provide a method to produce the plugin description.
- Submit his plugin for Registration.

The DEAL - Driver Management System

- Storage Device(s) + Execution Environment = Deployment Environment.
- Different Deployment Environments can be more suited for a specific task (e.g. an MLP TEST is unlikely to be a computing intensive task, so GRID latency times are unnecessary).
- Dynamic Driver Loading $\longrightarrow$ Driver Plugins.
- Drivers are available to the Framework WS *and* to the Plugins.
- Also used to convert files among different formats (standard or DMM dependent).

The IT Developer perspective



If one wants to develop a new driver for his execution environment or storage system he just has to implement the Driver Plugin Interface and register it to the Driver Management System. We can provide the full specification and needed assistance.

Where to go now?

- Front End Extensions:
  - Tags
  - Interactive Visualization
- DMM/Methods engineering.
- Visualization methods engineering.
- Drivers Implementation:
  - Stand Alone (fallback, SDK, testing).
  - “European” Middleware (gLite) storage and execution.
  - ...
- More Web2.0:
  - Groups
  - Information Production/Sharing
- FW/DB Decentralization.

Acknowledgements

This work is partially funded by the Italian Ministry of Foreign Affairs.

So long and thanks for all the fish...