Mem. S.A.It. Suppl. Vol. 13, 111

Memorie della
Supplementi

The Future Astronomical Software Environment progress

L. Paioro[1], B. Garilli[1], P. Grosbøl[2], D. Tody[3,4], C. Surace[5], T. Fenouillet[5], P. Franzetti[1], M. Fumana[1], and M. Scodeggio[1]

[1] Istituto Nazionale di Astrofisica – Istituto di Astrofisica Spaziale e Fisica Cosmica, Via Bassini 15, I-20133 Milano, Italy e-mail: `luigi@lambrate.inaf.it`
[2] European Southern Observatory, Karl-Schwarzschild-Strasse 2, Garching, Germany
[3] National Radio Astronomy Observtory, Socorro, NM, U.S.A.
[4] U.S. National Virtual Observatory
[5] Laboratiore d'Astrophysique de Marseille, Traverse du Siphon, 13376 Marseille, France

Abstract. The OPTICON working group 3.6 in collaboration with international partners and in coordination with the Virtual Observatory, has already identified the high level requirements and the main architectural concepts for a future software environment for astronomical data reduction and analysis (Future Astronomical Software Environment). A special attention has been payed to: a) scalability, to allow the reduction of huge data volumes exploiting the hardware and software parallel architecture, b) interoperability, in order to guarantee the interaction between software coming from different sources and make easy the access to the Virtual Observatory, c) and modularity, to separate the adopted software technology from the specific computational algorithm and allow an independent evolution of the two areas.
The proposed concepts have been widely discussed and shared by the astronomical community; however a lot of work still remains to do, mainly: a) the definition of open standards, b) the verification of such standards thanks to at least one reference implementation and practical user cases, c) and the whole must be supported at least by the major international organizations that develop data reduction and analysis software.
All this work has led up to the definition of a new proposal for FP7 within OPTICON (where ESO, INAF, LAM-OAMP and NRAO/NVO are actively involved) which we present describing the project in detail and adding a description of the European FASE prototype, developed by INAF-IASF Milano in collaboration with LAM-OAMP (Marseille).

Key words. astronomy – data reduction – data analysis – Virtual Observatory – parallel computing – software environment

1. Introduction

In the following paper we present the current working status and the plans for the next development of a new software environment for astronomical data reduction and analysis (the Future Astronomical Software Environment).

The high level requirements and the main architectural concepts have already been de-

Send offprint requests to: L. Paioro

fined within the OPTICON working group 3.6 in collaboration with international partners such as NRAO/NVO (Tody et al. 2006). This basilar initial work has been carried on thanks to a specific FP6 funding.

A new proposal for FP7 has alredy been submitted to OPTICON and will be addressed to the European Union with the purpose of receiving fundings for at least one practical implementation and for the consequent architectural design refinement.

2. What astronomers use

For decades the astronomical community has used several data reduction and analysis software which perfectly fitted their scientific necessities. Good examples of such software are: IRAF, MIDAS, AIPS, GIPSY and so on, which are still very important tools for data handling, widely used as today as in the past.

However, their age and resulting technological backwardness make them rather incompatible with the modern scripting languages commonly used nowadays. It is also fairly complicated to share pieces of code or data among them and there is scarce support to the new distributed technologies like Virtual Observatory, GRID, MPI/HPC, etc.

3. What astronomy need

New frontiers of science require collecting as much information as possible. This implies the handling of huge amounts of data products and/or simulated data, usually shared by geographically distributed research groups. In general, we can identify four user cases which well depict the present scientific and technological needs that future hardware and software tools must support:

- *Single analyst user on local workstation*. He or she needs an integrated environment that allows to analyse and in case reduce the data products he or she owns. Such an integrated environment must be able to expose to the end user a large variety of tools (including those made by the user) through a common framework, letting the user to get rid of the burden of data format conversions and all those issues which arise when dealing with different tool from different providers, usually rather incompatible one with another.
- *Team of big survey analysts*. They have certainly the same needs of the single analyst user, but they also need a shared and collaborative environment which is geographically distributed and provides high performance computing power.
- *Single algorithms developer*. One of the most common aspect of the astronomical research is the development of more and more new algorithms for data analysis and theoretical simulation. Unfortunately, the creative effort addressed to invent new algorithms must give way to the practical effort addressed to learn and apply how to implement such new algorithms at software level. This is an unhappy wasted time that requests tools to simplify the development of new computational tasks splitting the technical competences from the inventive ones.
- *Team of software developers*. The cost of a new data reduction and analysis software development is always made bigger by the necessity of starting from scratch redesigning and building up very basic and common procedures and/or data structures every time. A common development framework which provides a basic software layer such as a shared distributed execution engine, a set of libraries suited around well defined data structures ad interfaces, one or more standard messaging system and a well supported set of bindings for all the most common programming languages (Java, Python, C/C++, FORTRAN, Perl, PHP, Tcl/Tk, .NET, etc.), in brief a complete development platform for astronomical applications, would be highly desiderable and actually already needed.

4. What F.A.S.E. is

Many of the issues above have already been faced up by the Virtual Observatory. On the other hand FASE project doesn't aim to be a

Virtual Observatory competitor but rather to start from it and create a new software platform for the astronomy with extended and even more extensible capabilities. The principal targets are:

a) the reuse of the most important legacy software within a modern framework that allows to extend their functionalities (see Fig. 1) and
b) make easier the support and development of new interoperable and distributed applications or simple computational tasks.

Besides these foundamental targets FASE aims to:

c) increase software sharing and astronomical software development collaboration;
d) support a simplified access to remote resources as archives, Virtual Observatory and GRID;
e) define stable, controlled and open software interfaces;
f) realize a first minimal reference implementation allowing future extended versions;
g) specify the basic free and open source system components, that could be well interfaced with commercial software as well.

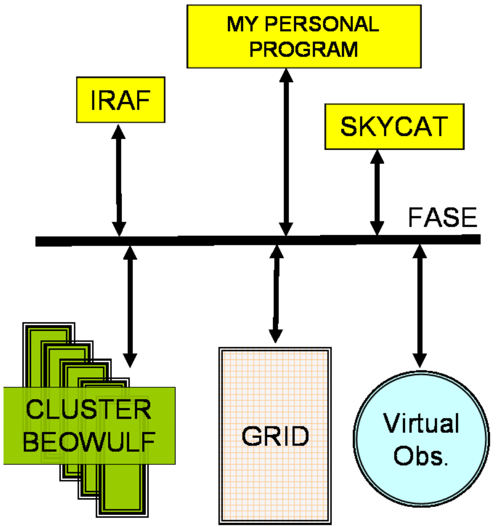

Fig. 1. FASE can be seen as a middleware that allows common astronomical software like IRAF, Skycat or astronomers' self-developed tasks to be interfaced with the new important distributed and interoperable technologies like Virtual Observatory, clusters and GRID.

5. The architecture

In order to achieve this ambitious goal, the architectural design foresees a modular approach (see Fig. 2). This way each single framework component (or set of components) can be developed apart and included in the whole framework as a plugglable module (or package).

The development of new framework modules is made easier thanks to the component container paradigm. A container is a sort of wrapper that connects any component (application or computational task) to the framework using standard, well defined and documented interfaces available for all the most common languages.

The main benefit of a modular architecture is the flexibility and extensibility. The basic framework capabilities can be expanded with new components added locally or distributed to the wide astronomical community.

The idea is to allow the addition of a number of modules developed by an open community as already happens in other well known communities like Mozilla Foundation[1], for instance. All the new public modules should be freely distributed through a central plug-in portal connected to the framework via a package/plug-in manager.

6. The spiral lifecycle model

The software development process adopted by OPTICON working group 3.6 to carry out the design and realization of the FASE environment is the spiral lifecycle model (Boehm 1988).

The spiral model is an evolution of the classic waterfall model (Royce 1970), which defines a monolithic sequence of operations: requirements collection, product design, implementation, verification and maintenance. The

[1] http://www.mozilla.org/foundation/

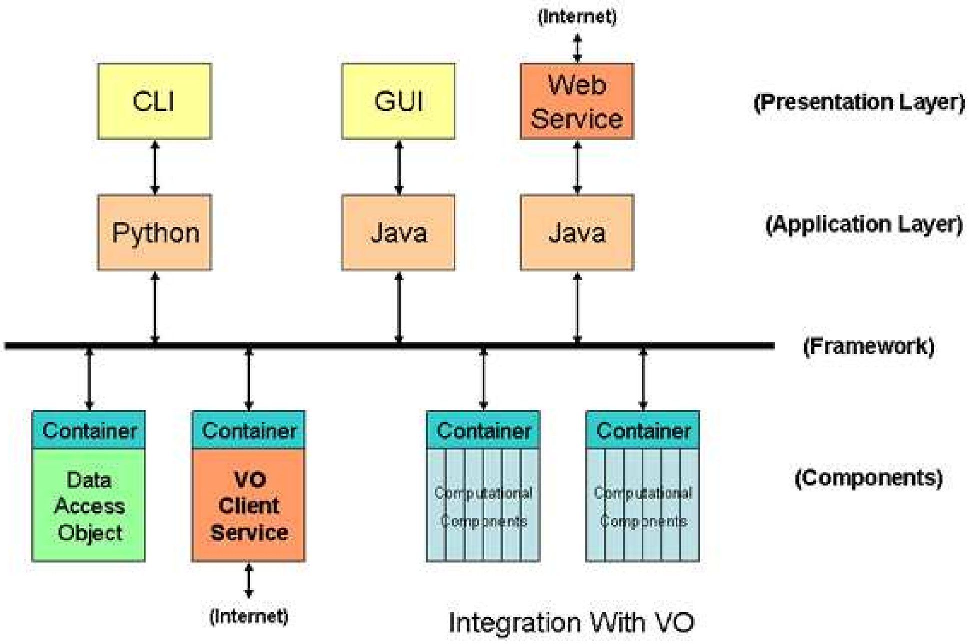

Fig. 2. The modular architecture of FASE. It can be schematized with four layers: the presentation layer (what the user see), the application layer (what is locally working), the framework (the distributed execution engine) and at the final endpoint the components which live within their containes.

spiral model combines the waterfall model with a prototyping approach, resulting more effective and suitable for large and complicated projects.

As shown in Fig. 3, the spiral model starts from the definition of general concepts which then must match with a set of requirements and user-cases. Just after these preliminary steps that must bring to a first software design, the prototyping is performed with the intention of producing feedbacks that can improve and refine the software design itself. In general this mutual interaction between design, development and prototyping is kept iterative along the whole software development process, up to the final software release.

So far, the preliminary steps have been almost completed producing a project TWiki site[2] that contains the minutes of the teleconferences done during the last five years, a set of pages and attachments containing the first design concepts and the detailed user-cases.

The most important resulting product of this preliminary work is the High-Level Requirement Document v1.00 (Grosbøl, Garilli et al. 2008), which defines in detail the requirements for:

1. the framework installation and run time processing;
2. the scripting language and execution environment;
3. the data access and data handling;
4. the applications lifecycle and development.

Following the spiral model philosophy, almost contemporary to the production of these documents, the development of two framework prototypes has been stared.

The USA NVO/NRAO prototype is mainly addressed to the Virtual Observatory access integration, and in order to achieve this goal it

[2] http://archive.eso.org/opticon/twiki/bin/view

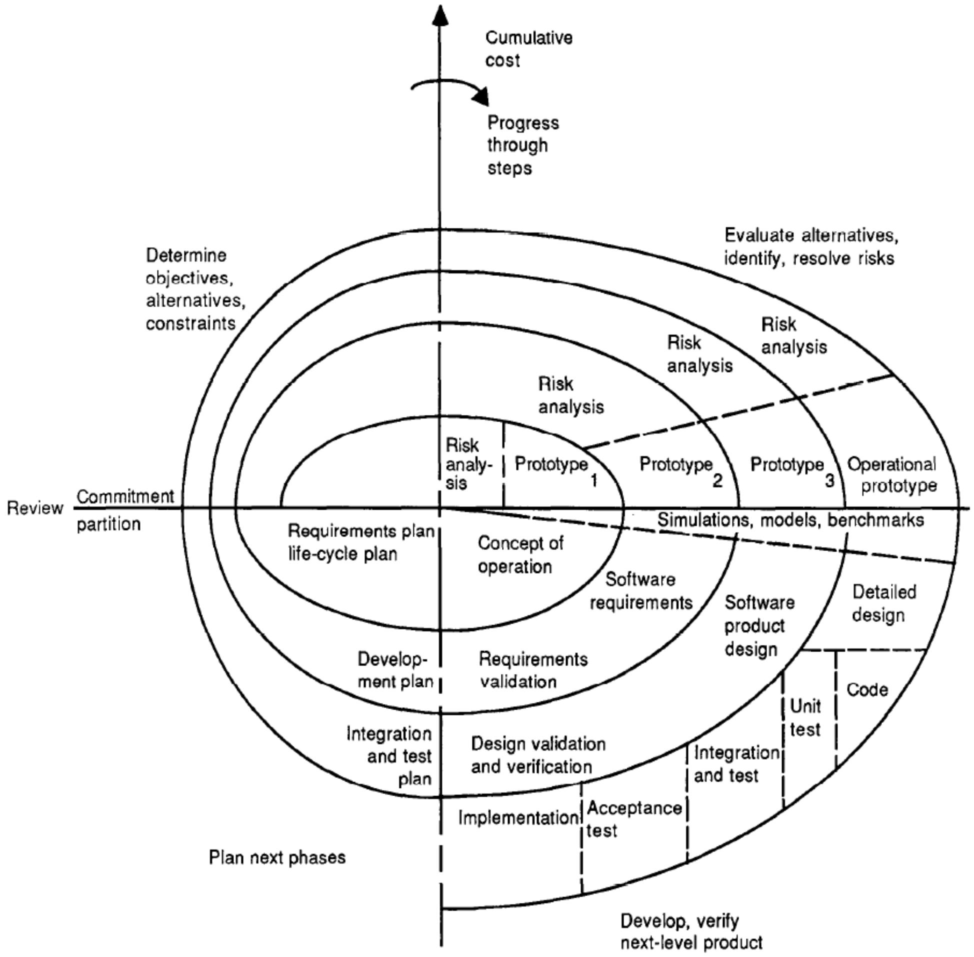

Fig. 3. Spiral model of the software process as originally described by Boehm (1988).

is focused around the NVO VOClient[3] application. The technologies adopted are mainly SAMP/PLASTIC protocol (Boch et al. 2009) at host level and OpenRTE (Castain et al. 2005) at inter-host level.

The European prototype is developed by INAF-IASF Milan in collaboration with LAM. This prototype is mainly focused on the task launching system development and is being carried on using the VIPGI recipes (Scodeggio et al. 2005) as test-bench. In fact VIPGI recipes are compiled programs that perform data reduction tasks based on sets of parameters. This pattern has been generalized as a global task/parameter paradigm that actually addresses many of the user needs described in paragraph 3.

The present European prototype uses DBus[4] as messaging system but it is planned to be moved to SAMP/OpenRTE in order to be fully integrated with the USA prototype in the near future. A complete documentation can be obtained visiting the project Trac site[5].

As a result of the work described above, the next step in the spiral model the is the prepara-

[3] http://iraf-nvo.noao.edu/vo-cli/

[4] http://www.freedesktop.org/wiki/Software/dbus

[5] http://cosmos.iasf-milano.inaf.it/trac/fase

tion of a detailed architectural document and a document which defines the system interfaces.

7. Future developments

As mentioned above, OPTICON working group 3.6 is going to submit a proposal for FP7 to the European Union in order to obtain funding to make more tangible what is still mainly theoretical (with the exception of the prototypes).

The proposal is meant to produce the following deliverables:

1. Steering Committee Meetings
   (a) Face-to-face meetings;
2. Open Standards Design
   (a) Software designer and engineer;
   (b) Support for travels for meetings etc.;
3. Reference Implementation Development
   (a) Execution framework;
   (b) Parameter management;
   (c) Application language bindings;
   (d) User interface connection;
   (e) VO and Web Services;
   (f) Access to legacy tasks.

If obtained, these fundings would be used by the European institutes involved in the project to actively contribute to the framework development, sharing the commitment with the USA partners.

References

Boch, T. et al. 2006, PLASTIC - a protocol for desktop application interoperability, Version 1.00, IVOA Note 01 June 2006. See also http://www.ivoa.net/cgi-bin/twiki/bin/view/IVOA/ApplicationsMessaging for discussions regarding SAMP

Boehm, B.W. 1988, A Spiral Model of Software Development and Enhancement, IEEE, 0018-9162/88/0500-0061

Castain, R.H. et al. 2005, The Open Run-Time Environment (OpenRTE): A Transparent Multi-Cluster Environment for High-Performance Computing, Proceedings, 12th European PVM/MPI Users' Group Meeting, available at http://www.open-rte.org/papers/euro-pvmmpi-2005-orte/

Grosbøl, P., Garilli B.M. et al. 2008, High-Level Requirements for a Future Astronomical Software System, available at the OPTICON Network 3.6 Twiki site (see reference in paper)

Royce, W. 1970, Managing the Development of Large Software Systems, Proceedings of IEEE WESCON 26 August, 1-9

Scodeggio, M. et al. 2005, The VVDS Data-Reduction Pipeline: Introducing VIPGI, the VIMOS Interactive Pipeline and Graphical Interface, PASP, 117, 1284

Tody, D. et al. 2006, An Open Architecture and Framework for Astronomical Data Processing and Analysis, PASP, 351, 331