J Grid Computing (2016) 14:673–685
DOI 10.1007/s10723-016-9382-y

# Web Services as Building Blocks for Science Gateways in Astrophysics

**S. Sánchez-Expósito · P. Martín · J. E. Ruiz · L. Verdes-Montenegro · J. Garrido · R. Sirvent · A. Ruiz Falcó · R. M. Badia · D. Lezzi**

Received: 28 January 2016 / Accepted: 19 October 2016 / Published online: 26 November 2016

**Abstract** An efficient exploitation of Distributed Computing Infrastructures (DCIs) is needed to deal with the data deluge that the scientific community is facing, in particular the Astrophysics one due to the emerging Square Kilometre Array (SKA) telescope that will reach data rates in the exascale domain. Hence, Science Gateways are being enriched with advanced tools that not only enable the scientists to build their experiments but also to optimize their adaptation to different infrastructures. In this work we present a method, called "two-level workflow system", to build this kind of tools and we apply it to a set of analysis tasks of interest for some use applications to the SKA. This method uses the Software-as-a-Service model to keep the scientists insulated from technical complexity of DCIs, and the COMPSs programming model to achieve an efficient use of the computing resources.

**Keywords** Astronomy · Virtual observatory · Web services · Workflows · Science gateways · Distributed computing infrastructures

S. Sánchez-Expósito (✉) · J. E. Ruiz · L. Verdes-Montenegro · J. Garrido
Instituto de Astrofísica de Andalucía - CSIC, Glorieta de la Astronomía s/n, 18008, Granada, Spain
e-mail: sse@iaa.es

J. E. Ruiz
e-mail: jer@iaa.es

L. Verdes-Montenegro
e-mail: lourdes@iaa.es

J. Garrido
e-mail: jgarrido@iaa.es

P. Martín · A. Ruiz Falcó
Fundación Centro de Supercomputación Castilla y León Edificio CRAI-TIC, Campus de Vegazana s/n. 24071, León, Spain

P. Martín
e-mail: pablo.martin@fcsc.es

A. Ruiz Falcó
e-mail: antonio.ruizfalco@fcsc.es

## 1 Introduction

The development and use of Distributed Computing Infrastructures (DCIs) has been mostly driven by the increasing demand for computational power in scientific applications. This has necessitated in many cases

R. Sirvent · R. M. Badia · D. Lezzi
Barcelona Supercomputing Center (BSC), Jordi Girona, 29, 08034, Barcelona, Spain

R. Sirvent
e-mail: raul.sirvent@bsc.es

R. M. Badia
e-mail: rosa.m.badia@bsc.es

D. Lezzi
e-mail: daniele.lezzi@bsc.es

adapting codes to the specific computing platform also forcing the scientists to be aware of technical characteristics of the platform [1]. This situation is far from ideal given the data deluge faced nowadays by the scientific community in general, and by Astrophysics in particular.

Science Gateways (SGs) have been developed with the aim of providing users with friendly tools that ease the interaction with computing infrastructures. They allow users to interact with applications ported to the DCIs, create workflows built upon these applications, reuse workflows shared by other users and access advanced data browsing and visualisation tools or portlets that ease the customization of applications. In these environments, two user categories can be distinguished: end-users and application developers. The latter are responsible for adapting scientific software to the DCI and providing the former with the necessary blocks to build their use cases. Application developers deal with scientific applications that are sequential in some cases and that are not optimized to profit from the capabilities of DCIs. Tools like COMPSs [2] or Dispel4py [3] facilitate the exploitation of the inherent parallelism of these applications. They are able to discover dependencies among application tasks, submitting them to DCIs as concurrently as possible and with minimal transformation of the original code.

The framework WS-PGRADE/gUSE [4, 5] has been used to develop several Science Gateways, such as MoSGrid [6] in the field of Life Science, and VisIVO [7] and CTA Science Gateway [8] in Astrophysics. In MoSGrid, Information Technology (IT) skilled users implement services and workflows for molecular simulations which can be reused by scientists to build their own experiments. VisIVO gathers a set of workflows that run on DCIs for visualising large-scale astrophysical datasets. It offers portlets to ease parameter-setting of workflows, allowing users to adapt them to their needs or use them as templates to build new ones. CTA Science Gateway provides a desktop environment that allows exploiting the graphical interface as provided natively by the tools included in this SG. The [1]BioVeL project provides a different approach. It exploits the advantages of the Software as a Service (SaaS) model, implementing a set of web services that interface several biological applications and that can be combined by end-users using the workflow management system Taverna Workbench [9]. BioVeL users can also share their workflows through the myExperiment portal [11] or the [2]BiodiversityCatalogue. In the Astrophysics domain, the HELIO project [10] also follows this approach and provides this community with workflows that implement new and more complex functionalities.

In Astronomy, the [3]International Virtual Observatory Alliance (IVOA) promotes the development of the Virtual Observatory (VO)[4], a web-distributed interoperable data network using common standards for data publishing, discovery and sharing. Moreover, efforts are being undertaken inside IVOA to promote standards for computational service access and interoperability. IVOA standards are widely used by the Astronomy community to build web services for data management, known as VO services. This makes the development of Science Gateways in Astrophysics easier, taking advantage of available standardised datasets and tools.

The astronomical community is currently working on the design of the infrastructure for managing the heavy and complex datasets that the Square Kilometre Array (SKA) will generate. SKA is an instrument that once built, will be the world's largest scientific infrastructure on Earth. It will be completed around 2028, when it will be able to generate up to 10 exabytes of data per day. Therefore, SGs in Astronomy require high capacity computing and networking resources in order to empower data management services compliant with IVOA standards, as well as advanced tools for the reduction and analysis of large datasets. The SKA pathfinder projects are working on building more efficient pipelines to reduce the data - i.e. to transform the stream of raw data into science-ready data - applying Big Data techniques among others. An illustrative example involves a project running in the LOFAR (LOw Frequency ARray) radiotelescope. It stores data in an HPC cluster specifically designed to support data-intensive work. These data are accessed through a MonetDB database that accelerates an automated pipeline aimed towards detecting

[1]https://www.biovel.eu/

[2]https://www.biodiversitycatalogue.org/

[3]http://www.ivoa.net

[4]Note that the abbreviation VO is not for Virtual Organisation as commonly it is used

transient sources [12]. The Astro-WISE information system [13] provides an environment where LOFAR data are processed, stored and managed across heterogeneous computing resources including a Grid node. This environment can be considered as an SG focused on data reduction. However, effort is still required to improve the tools for analysing and visualising large volumes of science-ready data. Analysis and visualisation tools should be incorporated as well into environments where a wider user community can access them. Along this line we find CyberSKA [14], a collaborative web portal for radio astronomy that provides access to several data management and visualisation tools.

As part of our efforts to provide astronomers with advanced tools that facilitate the efficient exploitation of DCIs, in this work we have focused on those SG blocks dedicated to data analysis. Data management as well as pipelines for data reduction are beyond the scope of this paper. We have designed these SG blocks by adopting an approach, used in the Life Sciences domain where computing and scientific applications are made available *as-a-Service* as well as empowered by COMPSs to achieve an efficient exploitation of the computing resources [15]. COMPSs orchestrates in a workflow the tasks involved in the service, submitting them as concurrently as possible to DCIs. The COMPSs workflows are exposed as web services that can be combined by scientists through graphical workflow manager systems like the Taverna Workbench. The resulting method is presented as a two-level workflow system: at the user level, workflows are built upon web services, while those services turn out to be workflows as well at the infrastructure level. We have applied this method to the Astrophysics domain, using as example a set of analysis tasks of interest for some use cases of the SKA.

In this paper we describe in detail a proposed two-level workflow system (Section 2). We then describe the use case in the Astrophysics domain (Section 3), we detail the application of this system to this particular use case (Section 3.2) and review the collection of implemented services for the analysis of interferometric data (Section 3.3). We refer to them as AMIGA4GAS services since they have been developed within the framework of the [5]AMIGA for GTC, ALMA, and SKA pathfinders project (AYA2011-30491-C02), whose technical objective was to build a federated layer that facilitates for astronomers the execution of workflows in heterogeneous DCIs. We significantly extend Sanchez-Exposito et al. [16] by detailing the design of the COMPSs applications and their deployment on different DCIs as well as assessing the advantages and challenges of the two-level workflow system (Section 4). Finally, we present our main conclusions in Section 5.

[5]http://amiga.iaa.es/p/263-federated-computing.htm

## 2 Two-Level Workflow System

This system addresses both the need of keeping scientists unaware of the technical complexity behind DCIs and the problem of exploiting computing resources efficiently. With this aim, on the one hand it adopts the SaaS approach, which enables scientists to focus on the experiment, releasing them from tasks like installing and updating the software, or configuring the execution framework. And on the other it uses the COMPSs programming model, which exploits the inherent parallelism of the applications in order to achieve an efficient use of the computing resources.

In this system, the tasks involved in a scientific analysis are transformed into web services, making their orchestration possible through any workflow manager system that supports web services (e.g. the Taverna Workbench). These web services are building blocks that end-users can combine to create scientific workflows, which are called user-level workflows.

Furthermore, a web service can perform one or several analysis tasks. They are orchestrated by COMPSs, which builds their dependency graph in order to submit them to the DCIs as concurrently as possible. Therefore, web services internally execute COMPSs-based workfows, which are called infrastructure-level workflows.

### 2.1 Architecture

Figure 1 depicts the architecture of the two-level workflow system. From the end-user point of view the infrastructure-level workflows are single web services that can be combined to build scientific workflows. End-users would import them into workflow manager supporting web services, and connect them to other services or tools by defining the data dependency between them through the mechanism provided by the workflow manager. Internally these web services invoke COMPSs that orchestrates the execution of the analysis tasks and submits them to a computing platform defined by the *Federation Layer*. This layer reads the service input parameters, including the platform where the dataset/s to be analysed is/are located. Then, it decides the platform where the tasks will be submitted, based on a set of business rules that take into account factors like the cost of moving datasets to other platforms or the platform availability, among others. Once the computing platform is selected, this layer configures the COMPSs framework by providing details about the platform, e.g. the Workload Management System (WMS) URL in case of a Grid infrastructure. The *Federation Layer* also gathers the user identity (e.g. a pair of user name and password), matches it with the identity

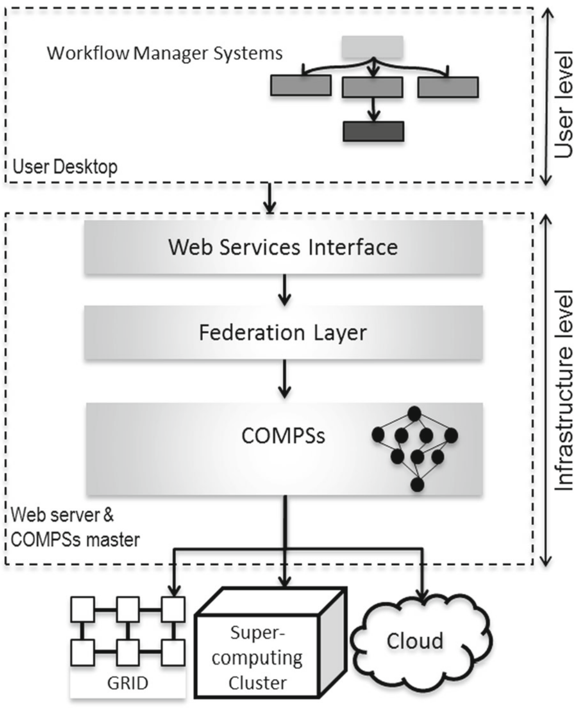


**Fig. 1** Two-level workflow system architecture. The user-level workflows are built upon web services that in turn are infrastructure-level workflows. Internally these web services invoke COMPSs that submits the tasks to a computing platform previously defined by the *Federation Layer*

format required by the computing platform (e.g. a X.509 certificate for Grid platforms), and provides the COMPSs framework with this user authentication information.

The goal of the *Federation Layer* is twofold: to facilitate user identification and to select the most suitable infrastructure for running the tasks depending on a set of factors. For the former, the *Federation Layer* uses the OASIS Security Assertion Markup Language to provide single sign-on identification, and robot certificates to prevent users from creating or using their own certificates. For the latter, the business rules system quantifies how suitable each computing infrastructure is for the current execution. This is done by means of a mathematical equation, which is composed of factors inherent to the user (e.g. access permissions), to the task (e.g. size of the input parameters) and to the computing resource (e.g. resource availability). Further details about the *Federation Layer* will be presented in [17].

The next layer in the two-level workflow architecture is COMPSs. In this framework application developers mark the methods invoked from their sequential application to be run as remote tasks on the available resources. COMPSs provides a mechanism based on Java annotations (or Python decorators) that enables application developers to indicate task parameters. This model ensures that data dependencies between tasks will be dynamically detected and enforced when running those tasks. Tasks can share data through their parameters, which can be objects, arrays, files and primitive types. The way tasks access such data defines their dependencies and the inherent concurrency of the application, which is automatically exploited by COMPSs. The availability of different connectors, each implementing the specific provider API (e.g. Cloud providers), makes it possible to run computational loads on multiple backend environments without the need of code adaptation. COMPSs also provides elasticity features in cloud environments that allows to adapt dynamically the number of utilized resources to the actual execution needs. The maximum and minimum number of required VMs can be specified in the COMPSs configuration. This enables the *Federation Layer* to indicate the needed computing resources based on the information about the service. For example, if the service groups $n$

parallel series of tasks, the *Federation Layer* would configure COMPSs to ask for *n* VMs at least.

### 2.2 Web Service Granularity

In the two-level workflow system, the tasks involved in an analysis process are grouped into several infrastructure-level workflows, which are exposed as single web services. The service granularity is defined as the number of analysis tasks executed in the service.

Different requirements determine how to group analysis tasks. A scientific workflow composed of web services of low granularity allows the user to combine the services in different ways, providing more flexibility to build new experiments. In the extreme case where a single web service would group all of the tasks involved in the experiment, this service would be seen as a black box by the end-users since, they will surely not be able to adapt it to their needs (by e.g. replacing a task or performing operations on the intermediate data). In general, those tasks that are usually connected in a straightforward way, with no manipulation of intermediate data by the end-user, are candidates to be grouped into a service. On the contrary, tasks should be grouped into different services, if the end-user builds different experiments by modifying the operations on their intermediate data.

Series of tasks whose intermediate datasets are significantly large, should be also grouped into the same service. This avoids data movement between computing platforms since the tasks orchestrated in the infrastructure-level workflows are always executed in the same platform. Therefore, a greater granularity in the service may reduce its execution time not only because it benefits from COMPSs' orchestration capability but also because there is no extra overhead due to the communication of intermediate data between tasks. This statement is also supported by the performance evaluation of services with different granularity described in Section 3.3.

Services that output large datasets should return references to them instead of the complete datasets. In this way, services receiving these references will get the computing platform where the input datasets are located, so that their *Federation Layer* can decide to submit their tasks to this platform.

## 3 Science Use Case

### 3.1 Kinematical Modelling of Galaxies

One of the SKA science drivers is the study of galaxy evolution, where the study of the atomic gas (HI) plays a key role. Radio interferometers like the SKA generate data with two spatial coordinate axes and a spectral axis containing information about the atomic gas distribution and velocity. Astronomers analyse these HI datacubes to produce a kinematical model of the galaxies, key to track perturbations or define the dynamical structure and hence the distribution of matter in galaxies.

The kinematic modelling of a galaxy can be performed by means of the so-called tilted-ring model [18], which consists in simulating the observed HI kinematics of the galaxy disk as a set of circular, differently inclined and positioned rotating rings. Various software packages exist that allow astronomers to fit a tilted-ring model to a velocity cube. The Groningen Image Processing System (GIPSY) [19] is one of them. It was designed for the reduction of Westerbork Synthesis Radio Telescope data. It has evolved since its creation and now it is able to handle data from many instruments.

The experiment implemented in this paper involves three GIPSY tasks: ROTCUR, ELLINT and GALMOD. ROTCUR derives the kinematical parameters of a galaxy, including the rotation curve; ELLINT generates the radial profile of the galaxy emission taking as input the results provided by ROTCUR; GALMOD builds a model cube from the kinematical and geometrical parameters, as well as the emission profile issued by ROTCUR and ELLINT respectively.

Users provide a set of input data, some of them compiled from external catalogues or derived by visual inspection of the data. It is a common practice to test different values of them by sweeping a small range, so as to generate several models. Then the astronomer assesses which model fits the original data better. This use case is shown in Fig. 2.

The here implemented services have taken into account the requirements of future SKA users. The experience from the SKA pathfinder LOFAR suggests that the copious data flux that the SKA will

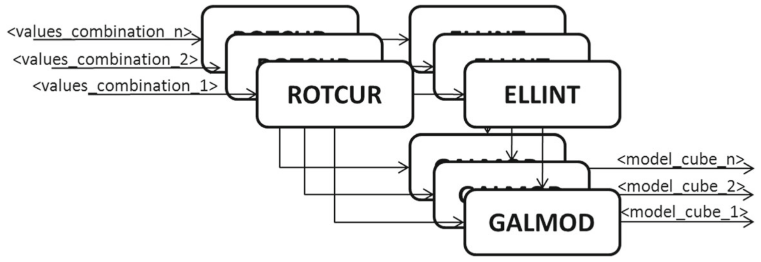


**Fig. 2** Use case schema. In order to build a model, the GALMOD task needs the kinematical parameters and the radial profile of the galaxy emission, provided by the ROTCUR and ELLINT tasks, respectively. In this schema, a parameter space of *n* combinations is explored, what requires *n* workflow executions

generate will need to be distributed to an heterogeneous computing infrastructure. In this situation, a uniform access will be needed to facilitate astronomers the exploitation of computing resources. In addition, the SKA will generate individual data products of several petabytes, what will require minimization of data transfer. In order to face this challenge, the implemented services have been connected to diverse computing resources, allowing the processing tasks to be executed where the data are stored and providing an homogeneous way of accessing them.

## 3.2 Implementation of the Two-Level Workflow System for Kinematical Modelling of Galaxies

Next we detail the process of applying the two-level workflow system to the use case of the kinematical modelling of galaxies. The initial Sections (3.2.1, 3.2.2 and 3.2.3) describe the implementation of the layers corresponding to the infrastructure-level workflows, also referred to as AMIGA4GAS services, starting with the porting of the scientific software to the computing platforms and continuing with the implementation of the COMPSs applications and the design of the web service interface. Then, we focus on user-level workflows in Section 3.2.4.

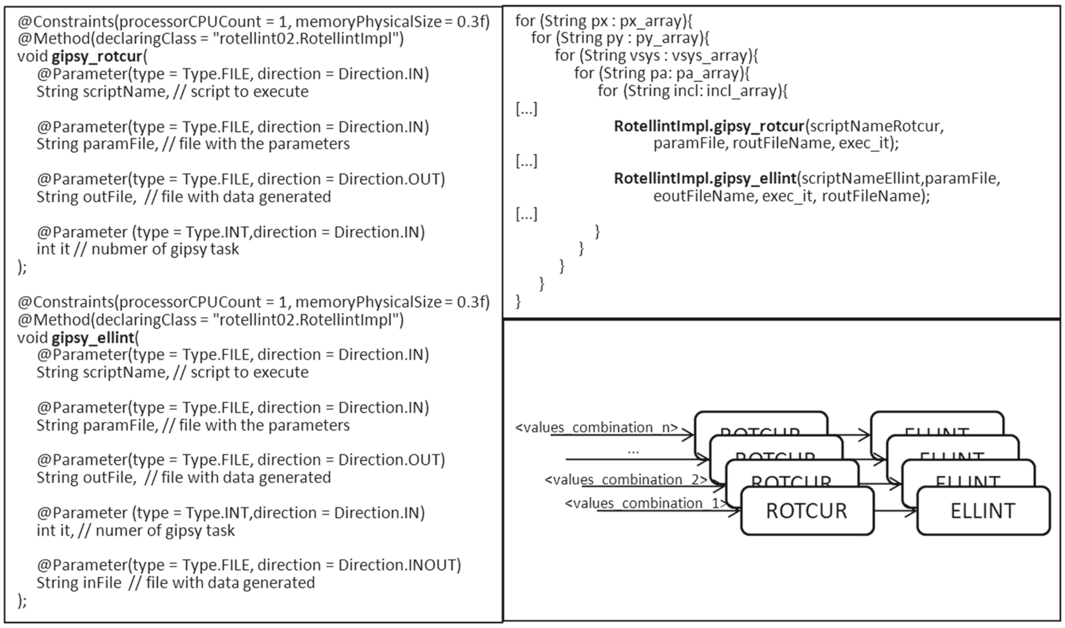
### *3.2.1 Deployment of GIPSY on Different DCIs*

GIPSY has been deployed on the supercomputing cluster provided by FCSCL, which implements an Open Grid Scheduler queuing system. GIPSY has also been ported to IBERGRID platform, the umbrella under which both the national Spanish and Portuguese Grid initiative operate. More specifically it was deployed on the IBERGRID sites that support the virtual organisation phys.vo.ibergrid.eu. To do this, GIPSY was first installed on a virtual machine (VM) with the same operating system and software libraries expected in a Grid working node. Then, the installation directory was packed in a tarball to be distributed by means of the CERN Virtual Machine File System (CernVMFS) [16], a tool for deploying software in distributed environments.

### *3.2.2 Implementation of the Applications with the COMPSs Framework*

The COMPSs runtime system distinguishes two kinds of nodes: the master node and the worker nodes. The former executes the main COMPSs application, and is in charge of detecting the invocation of the application tasks, building their dependency graph, launching them following the order determined by the graph and transmitting/gathering the required files to/from the location where the task execution takes place. The worker nodes (also known as workers) wrap the application tasks and are executed on the computing resource, performing the wrapped application task itself and the communication with the master node.

When one of the AMIGA4GAS services is called, the master executes the COMPSs application corresponding to the service. Figure 3 depicts part of the COMPSs application for the service rotellint_ws, which executes two GIPSY tasks, ROTCUR and ELLINT, to calculate the kinematical parameters and the emission radial profile of a galaxy. In particular, the left panel shows the code where the COMPSs tasks are declared; the upper right panel shows the code



```
@Constraints(processorCPUCount = 1, memoryPhysicalSize = 0.3f)
@Method(declaringClass = "rotellint02.RotellintImpl")
void gipsy_rotcur(
    @Parameter(type = Type.FILE, direction = Direction.IN)
    String scriptName, // script to execute

    @Parameter(type = Type.FILE, direction = Direction.IN)
    String paramFile, // file with the parameters

    @Parameter(type = Type.FILE, direction = Direction.OUT)
    String outFile,  // file with data generated

    @Parameter (type = Type.INT,direction = Direction.IN)
    int it // nubmer of gipsy task
);

@Constraints(processorCPUCount = 1, memoryPhysicalSize = 0.3f)
@Method(declaringClass = "rotellint02.RotellintImpl")
void gipsy_ellint(
    @Parameter(type = Type.FILE, direction = Direction.IN)
    String scriptName, // script to execute

    @Parameter(type = Type.FILE, direction = Direction.IN)
    String paramFile, // file with the parameters

    @Parameter(type = Type.FILE, direction = Direction.OUT)
    String outFile,  // file with data generated

    @Parameter (type = Type.INT,direction = Direction.IN)
    int it, // numer of gipsy task

    @Parameter(type = Type.FILE, direction = Direction.INOUT)
    String inFile  // file with data generated
);
```

```
for (String px : px_array){
   for (String py : py_array){
       for (String vsys : vsys_array){
           for (String pa: pa_array){
               for (String incl: incl_array){
[...]
                   RotellintImpl.gipsy_rotcur(scriptNameRotcur,
                           paramFile, routFileName, exec_it);
[...]
                   RotellintImpl.gipsy_ellint(scriptNameEllint,paramFile,
                           eoutFileName, exec_it, routFileName);
[...]
               }
           }
       }
   }
}
```

**Fig. 3** COMPSs application corresponding to the service rotellint_ws. The *left panel* shows the code where the COMPSs tasks are declared; the *upper right panel* shows the code of the main application where those tasks are called; the *lower right panel* shows the dependency graph generated by the COMPSs runtime

of the main application where those tasks are called; the lower right panel shows the dependency graph that is generated by the COMPSs runtime during the execution. ROTCUR is executed *n* times, where *n* is the amount of input parameters combinations. Then ELLINT is executed using as input the results of ROTCUR.

A set of two configuration files provides the COMPSs master with the characteristics of the computing platform where application tasks should be launched. Our implementation holds two sets of configuration files, one for each infrastructure where the GIPSY tasks can be launched: IBERGRID and the supercomputing cluster at FCSCL. The COMPSs master will use one configuration file set or the another, depending on which platform has been chosen by the *Federation Layer*.

### *3.2.3 Design of the Services Interface*

The AMIGA4GAS services act as the interface between the Taverna Workbench and the GIPSY tasks. Those tasks have a large number of input parameters, allowing them to perform several kinds of operations. Handling this amount of parameters in a graphical workflow manager such as the Taverna Workbench can be tedious. Reducing the number of input parameters by setting some of them with a default value constrains the specific operation that the service will perform. This would help non experienced GIPSY users, since they would not get confused by those parameters that they do not need. However, expert GIPSY users may miss some functionalities. Therefore, once the input parameters related with a graphical/interactive mode of the tasks were discarded, we took the decision to focus the services on the operations related to the kinematical modelling of galaxies, trying to find a compromise solution between flexibility and usability. We may still incorporate on demand new services that would implement more advanced operations to fulfil the requirements coming from expert users.

Astronomers usually explore different combinations of values for certain input parameters that are difficult to estimate with accuracy. This repetitive and time consuming job has been automated to cope with the large amount of expected data. Therefore, some AMIGA4GAS services have been implemented to receive both a single value and a range or a list of values for these parameters and execute the specific GIPSY task as many times as the possible

**Table 1** List of the main AMIGA4GAS Services

| WS Name | GIPSY Tasks | Outputs |
|---|---|---|
| rotcur_ws | ROTCUR | Kinematical parameters of the galaxy |
| ellint_ws | ELLINT | Radial profile of the galaxy emission |
| galmod_ws | GALMOD | A datacube modelling the input one |
| rotellint_ws | ROTCUR ELLINT | Kinematical parameters and radial profile |
| modelling_ws | ROTCUR ELLINT GALMOD | Kinematical parameters, radial profile and a datacube model |

combinations. Notice that users can also build a workflow in the Taverna Workbench that iterates over a list of values, calling and executing a low granularity atomic service many times in a loop to sweep the range of values. However, we have observed (Section 3.3) that the time execution of this Taverna workflow may be higher since the latency time of invoking the web service plus the time of receiving its output slows down the whole execution.

#### *3.2.4 Implementation of User-Level Workflows for the Kinematical Modelling of Galaxies*

The user-level workflows were built upon the AMIGA4GAS services, combining them in the Taverna Workbench, so as to implement different experiments that can be used as templates. In particular, we used the Taverna Workbench Astronomy 2.5, a special edition of the Taverna Workbench that includes support for building and executing astronomy workflows based on VO services through the Astrotaverna plug-in [20], which we have developed in the past. This plug-in integrates existing VO services as first-class building blocks in Taverna workflows. Besides, it provides this workflow manager with astronomical data manipulation tools.

### 3.3 AMIGA4GAS Services and Workflows for the Kinematical Modelling of Galaxies

The AMIGA4GAS services can be imported in the Taverna Workbench through a [6]specific URL. The access to the services is authenticated through a basic user/password mechanism. The [7]AMIGA4GAS services website contains detailed descriptions of the services as well as user and programming manuals.

Table 1 lists the main AMIGA4GAS services, the involved GIPSY tasks and their outputs. The services rotcur_ws, rotellint_ws and modelling_ws admit a range or list of input values, obtaining as many outputs as possible combinations of these values.

A set of Taverna workflows has been implemented using these services. Table 2 lists them, together with the web services involved in each workflow and the myExperiment ID that can be used to access them in the [8]myExperiment digital library.

The workflows rotellint_wf and rotcur_ellint_wf procure the same functionality. The difference between them is that they are built upon services with different level of granularity. While the former contains a single web service interfacing two GIPSY tasks, the latter makes use of two services, each one implementing one single GIPSY task. The same applies to modelling_wf and rotcur_ellint_galmod_wf.

Figures 4 and 5 represent the workflows rotellint_wf and rotcur_ellint_wf respectively. For the sake of clarity, these workflows are shown without the list of inputs. These two workflows represent different approaches to tackle with the same experiment. Low granularity web services allow the users to deal with intermediate data and provide them with more flexibility. High granularity web services provide a better performance in terms of execution time.

The rotellint_wf and rotcur_ellint_wf workflows are formed by one or two AMIGA4GAS services (vertical lines pattern boxes), Taverna tools for extracting

[6]https://srv-prj-wsamiga.fcsc.es:8444/Amiga4GasServiceLadon/soap11/description

[7]https://srv-prj-wsamiga.fcsc.es/

[8]http://www.myexperiment.org/workflows/$⟨$ID$⟩$.html

**Table 2** List of Taverna workflows

| Wf Name | WS involved | myExp. ID |
|---|---|---|
| rotcur_wf | rotcur_ws | 4609 |
| rotellint_wf | rotellint_ws | 4611 |
| rotcur_ellint_wf | rotcur_ws, ellint_ws | 4610 |
| modelling_wf | modelling_ws | 4613 |
| rotcur_ellint_galmod_wf | rotcur_ws, ellint_ws, galmod_ws | 4612 |
| vo_rotcur_wf | VO services and rotcur_ws | 4619 |
| vo_modelling_wf | VO services and galmod_ws | 4620 |

output data from web services (empty pattern boxes) and two nested workflows (black pattern boxes) for plotting the AMIGA4GAS service results. Dotted pattern boxes at the bottom represent the workflow outputs. Grey pattern boxes contain some constant values. In the rotellint_wf workflow, the rotellint_ws service executes internally the task ROTCUR, then processes its output data, calculating the average of the kinematical parameters of the central rings of the galaxy, and finally passes the resulting values to task ELLINT as input data. However, in the workflow rotcur_ellint_wf, since ROTCUR and ELLINT tasks are performed by different web services, the ROTCUR output processing is totally flexible. This is implemented through a Python script, embedded in the horizontal lines pattern box in Fig. 5, where users can select the galaxy rings from which they want to extract the kinematical parameters and perform different statistical operations on them (e.g. a median).

In order to confirm that a higher granularity improves service performance, we have compared two other workflows that only differ in the granularity of one of their services. We have built a new workflow called rotcur_ellint_list_wf, from the rotcur_ellint_wf workflow. In particular we have changed the rotcur_ws service configuring some of its input ports in such way they accept a list of values instead of a single value. This is what in the Taverna Workbench is called as configuring a service input port with *depth* 1. In this way, the Taverna Workbench will call the service as many times as possible combinations of the elements in the input lists. Each call will correspond to a single ROTCUR execution. Therefore, in this case, the granularity of the rotcur_ws service will be 1. In the initial rotcur_ellint_wf, the rotcur_ws input parameters were provided with single values. These values are character strings following a specific format for representing lists of values (e.g. numbers delimited by the ampersand character). In this way, the Taverna Workbench called the service only once, while the COMPSs application implementing the rotcur_ws service recognised the string format and executed *n* times

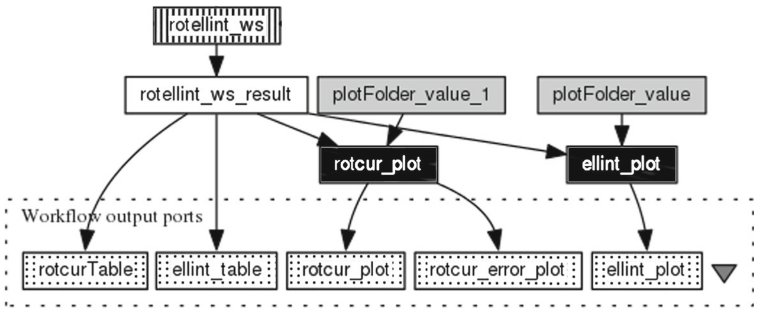


**Fig. 4** Taverna workflow rotellint_wf. This workflow calls the rotellint_ws service, which executes internally the task ROTCUR, then processes its output data, calculating the average of the kinematical parameters of the central rings of the galaxy, and finally passes the resulting values to the task ELLINT, as input data. Finally, two nested workflows (rotcur_plot and ellint_plot) will plot the rotellint_ws results

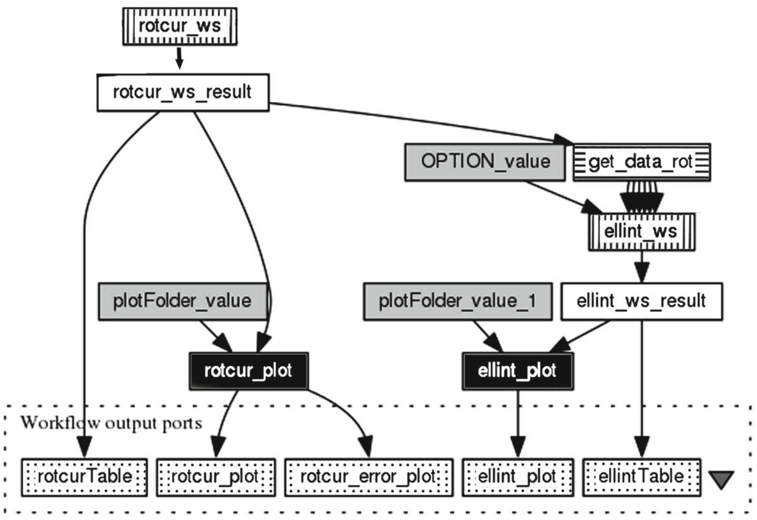


**Fig. 5** Taverna workflow rotcur_ellint_wf. This workflow calls the rotcur_ws service, which internally executes the task ROTCUR. Then, it gets the kinematical parameters of the galaxy by means of a python script (get_data_rot) that processes the rocur_ws results. These kinematical parameters are received by the ellint_ws service, which executes the task ELLINT. Finally, two nested workflows (rotcur_plot and ellint_plot) will plot the rotcur_ws and ellint_ws results

the ROTCUR task, where *n* is the number of possible combination of the input values. Therefore, while in rotcur_ellint_list_wf the iteration over the space parameter will be done by the Taverna Workbench, in rotcur_ellint_wf it was done by the corresponding COMPSs application.

We compared the completion time for rotcur_ellint_wf and rotcur_ellint_list_wf workflows. In order to avoid completion time variations due to the intrinsic platform performance, we predefined the FCSCL Cluster as the infrastructure where the tasks will be submitted by COMPSs. We executed the

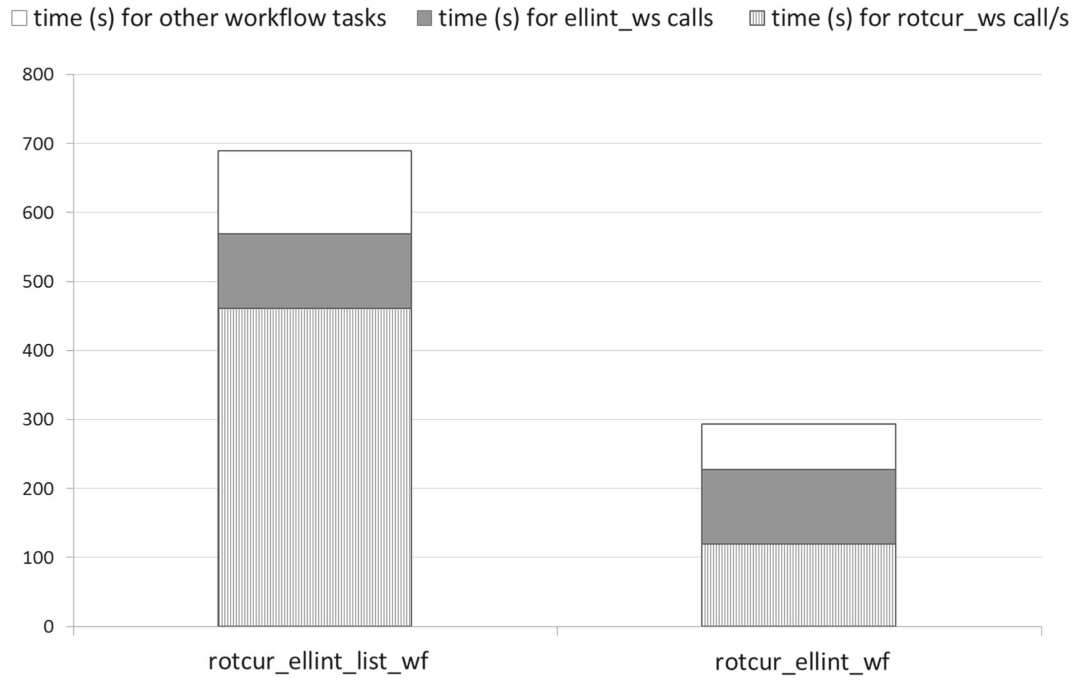


**Fig. 6** Diagram comparing the completion time of rotcur_ellint_list_wf and rotcur_ellint_wf. Both workflows are similar, except for the fact that in the former workflow the granularity of the rotcur_ws service is 1, while in the latter it is 81. The rotcur_ellint_list_wf workflow took about twice the time than the rotcur_ellint_wf workflow

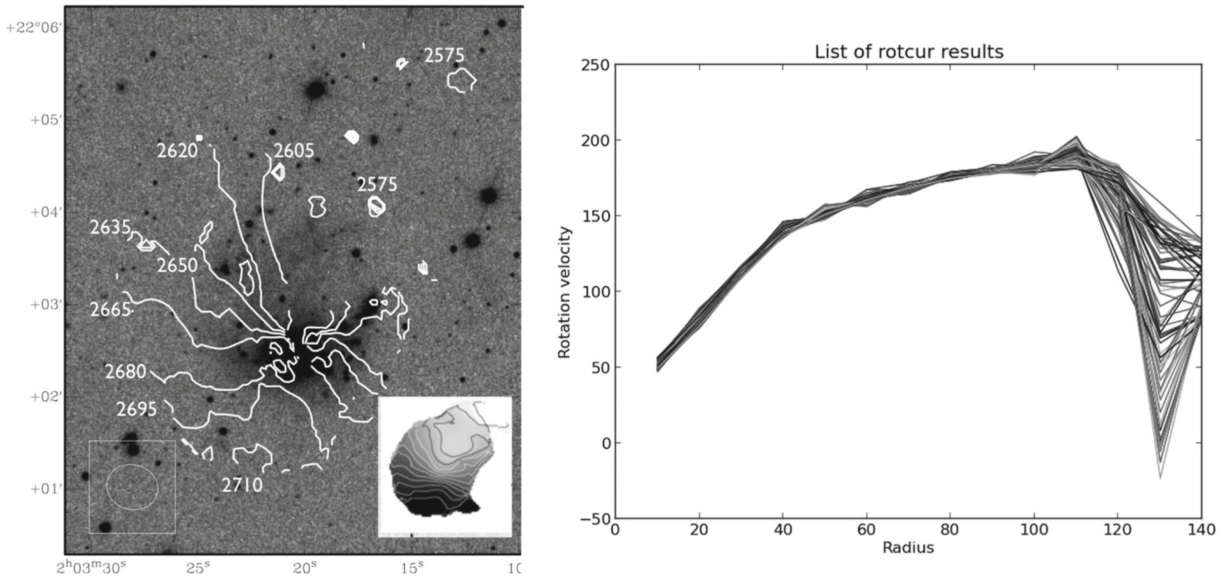


**Fig. 7** HI velocity field contours of the CIG85 galaxy and the results of exploring an input parameter range to calculate the rotation curve of this galaxy (rotcur_wf)

workflows in order to explore a parameter space of 81 combinations (4 input parameters of the rotcur_ws service received lists of 3 values). The completion time of these workflows is shown in Fig. 6. The rotcur_ellint_list_wf workflow took about twice the time than the rotcur_ellint_wf workflow, mainly due to the 81 web service calls performed in the former and that are managed internally by COMPSs at the infrastructure level in the latter.

We have also built workflows upon data discovery VO services. Specifically, the vo_rotcur_wf and vo_modelling_wf workflows integrate two VO services that are compliant with the [9]DataLink standard and the [10]SIAv2 recommendation. Both of them are IVOA data discovery protocols. DataLink provides additional metadata for datacubes discovered with SIAv2. The SIAv2 service queries a specific datacube archive, searching for datacubes inside a sky region specified by the user through e.g. a pair of coordinates and a search radius. Once the datacube/s fulfilling the search criteria are identified, the DataLink service delivers related complementary metadata, including the datacube/s location. We have used a specific representation of the dataset location in order to reference both image data and datacubes, since the size of this kind of data may be significantly higher than other service parameters.

Figure 7 contains several images illustrating the selected science case. The left panel shows an optical image of a galaxy (CIG85 [21]) with HI velocity field contours. Those contours are used to get some of the input values required by the rotcur_wf workflow. One of the results of executing this workflow is a set of 81 rotation curves, which are shown at the right hand panel of the figure.

## 4 Discussion

Domain applications are offered in a SaaS delivery model in our two-level workflow system. The scientific tasks provided by these applications are converted into web services, which are easily combined through graphical tools like the Taverna Workbench in order to implement scientific workflows. This enables scientists to focus on the scientific functionality of the applications, releasing them from technical tasks. However, the conversion of traditional software applications into web services may not be practical or useful in some cases. For example, the conversion of desktop visualisation tools may require excessive efforts. In other cases, the application functionality may be reduced. Specifically, the AMIGA4GAS web

[9]http://www.ivoa.net/documents/DataLink/
[10]http://www.ivoa.net/documents/SIA/

services are able to receive just a subset of the input parameters for the GIPSY tasks, setting the rest of them with a default value. This improves service handling in the Taverna Workbench, but it also constrains the GIPSY task functionalities.

Furthermore, the analysis web services take advantage of the COMPSs programming model in order to achieve a more efficient use of DCIs. COMPSs discovers the dependencies among tasks, so that it can execute them as soon as their input data are ready. Higher granularity services benefit from this COMPSs capability more than lower granularity services. However, services grouping several scientific tasks may be useless in use cases that require manipulation of the intermediate data according to end-users criteria. In addition, in this design, the latency due to the intermediate data transfer is reduced by 1) grouping into the same service series of tasks whose intermediate datasets are significantly large and 2) implementing services so that they return references to large output datasets instead of the datasets themselves. However, there are cases where it is not possible to avoid transferring large datasets. Specifically when an analysis service receives as inputs several references to large datasets located in different platforms. In these situations, the *Federation Layer* of the service will select the platform where the service tasks will be submitted and hence where the input datasets need to be transferred.

We have performed a preliminary evaluation of the performance gain of the granularity approach. However, more accurate and realistic estimations would be obtained by using a software application with computing requirements similar to those of the future tools for the analysis of SKA data. GIPSY is able to manage datacubes up to 8 GB, while datacubes generated by the SKA pathfinder LOFAR may reach several TBs, and the SKA datasets are expected to be larger.

## 5 Conclusions

Scientific communities are approaching the data deluge challenge through the use of Science Gateways where scientists can share and reuse tools that efficiently exploit DCIs for analysing and visualising large datasets. The astronomical community is paving the way to the SKA, an instrument that once built, will be the world's largest radio interferometer by far, able to reach exaescale data rates. Big Data techniques are being applied to improve the pipelines for the SKA data processing. However an effort is still needed in this field to improve the science-ready data analysis, through advanced services that both exploit efficiently the DCIs and keep astronomers unaware of the technical complexity behind these.

This work aims to contribute to these efforts with a two-level workflow system that at the lower level uses COMPSs to optimize the use of the computing resources, and at the higher level provides a web interface to ease the work of astronomers building their scientific workflows. This system has been applied to build a set of advanced tools that implement analysis tasks of interest for those SKA use cases aiming to perform kinematical studies of galaxies. At the infrastructure level, we designed a set of web services with different levels of granularity to be adapted to different user preferences. These services benefit from COMPSs to efficiently access computing infrastructures. At the user level, we developed a set of Taverna workflows built upon the aforementioned web services. They implement different use cases that can be used as templates, including workflows that combine kinematical analysis services with VO services. This system allows executing processing tasks on different kinds of computing infrastructures, minimizing in that way the data transfer and hiding the technical details of DCIs.

This work constitutes the first application of the two-level workflow system approach to tasks of interest for SKA. As future work we aim to apply this model to upcoming new software of interest for the analysis of SKA data. Indeed, it is being applied to a pipeline for calibrating [11]LOFAR data. The high computing requirements of the LOFAR software will allow getting more accurate estimations on the performance gain obtained by applying this model.

**Acknowledgments** This work has been developed in the framework of AMIGA4GAS (AYA2011-30491-C02) and AMIGA5 (AYA2014-52013-C2-1-R), co-funded by MICINN and FEDER funds. It has been also supported by the Wf4Ever STREP Project (270129) funded under EU FP7 (ICT-2009.4.1) and the Galaxias y Cosmologia project funded by the Junta de Andalucia (TIC-114). This work has also benefited from IBERGRID and the EGI Federated Cloud.

[11]http://amiga.iaa.es/FCKeditor/UserFiles/File/radio_fedcloud_v5_sinFondoA4.pdf

## References

1. Kiss, T., Kelley, I., Kacsuk, P.: Porting computation and data intensive applications to distributed computing infrastructures incorporating desktop grids inproceedings of science (2011)
2. Lordan, F., Tejedor, E., Ejarque, J., Rafanell, R., Álvarez, J., Marozzo, F., Lezzi, D., Sirvent, R., Talia, D., Badia, R.M.: Servicess: An Interoperable Programming Framework for the Cloud. Journal of Grid Computing **12**(1), 67 (2014)
3. Filguiera, R., Klampanos, I., Krause, A., David, M., Moreno, A., Atkinson, M.: Dispel4Py:A Python framework for data-intensive scientific computing. In: Proceedings of the 2014 International Workshop on Data Intensive Scalable Computing Systems, pp. 9–16 (2014). doi:10.1109/DISCS.2014.12
4. Kacsuk, P., Farkas, Z., Kozlovszky, M., Hermann, G., Balasko, A., Karoczkai, K., Marton, I.: WS-PGRADE/gUSE generic DCI Gateway Framework for a large variety of user communities. Journal of Grid Computing **10**(4), 601 (2012)
5. Balasko, A., Farkas, Z., Kacksuk, P.: Building Science Gateways by utilizing the generic WS-PGRADE/gUSE workflow system. Computer Science **14**(2), 307 (2013)
6. Krüger, J., Grunzke, R., Gesing, S., Breuers, S., Brinkmann, A., de la Garza, L., Kohlbacher, O., Kruse, M., Nagel, W.E., Packschies, L., Müller-Pfefferkorn, R., Schäfer, P., Schärfe, C., Steinke, T., Schlemmer, T., Warzecha, K.D., Zink, A., Herres-Pawlis, S.: The MoSGrid Science Gateway – A Complete solution for molecular simulations. J. Chem. Theory Comput. **10**(6), 2232 (2014)
7. Sciacca, E., Bandieramonte, M., Becciani, U., Costa, A., Krokos, M., Massimino, P., Petta, C., Pistagna, C., Riggi, S., Vitello, F.: VisIVO Science Gateway: A collaborative environment for the Astrophysics community. In: International Workshop on Science Gateways (2013)
8. Costa, A., Massimino, P., Bandieramonte, M., Becciani, U., Krokos, M., Pistagna, C., Riggi, S., Sciacca, E., Vitello, F.: An innovative Science Gateway for the Cherenkov Telescope Array. Journal of Grid Computing **13**(4), 547 (2015)
9. Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., Soiland-Reyes, S., Dunlop, I., Nenadic, A., Fisher, P., Bhagat, J., Belhajjame, K., Bacall, F., Hardisty, A., Nieva de la Hidalga, A., Balcazar Vargas, M.P., Sufi, S., Goble, C.: The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud . Nucleic Acids Research **41**(W1), W557 (2013). doi:10.1093/nar/gkt328
10. Le Blanc, A., Brooke, J., Fellows, D., Soldati, M., Pérez-Suárez, D., Marassi, A., Santin, A.: Workflows for heliophysics. Journal of Grid Computing **11**(3), 481 (2013)
11. De Roure, D., Goble, C., Stevens, R.: The design and realisation of the myExperiment Virtual Research Environment for social sharing of workflows. Futur. Gener. Comput. Syst. **561**, 25 (2009)
12. Scheers, B., Groffen, F.: Towards dynamic light-curve catalogues. In: Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 8451, p. 0 (2012). doi:10.1117/12.926069
13. Begeman, K., Belikov, A., Boxhoorn, D., Dijkstra, F., Holties, H., Meyer-Zhao, Z., Renting, G., Valentijn, E., Vriend, W.J.: LOFAR Information System. Futur. Gener. Comput. Syst. **27**(3), 319 (2011). doi:10.1016/j.future.2010.08.010
14. Kiddle, C., Taylor, A.R., Pigat, D., Eymere, O., Rosolowsky, E., Kaspi, V., Willis, A.G.: CyberSKA : An on-line collaborative portal for data-intensive radio Astronomy. In: ACM workshop on Gateway computing environments, pp. 65–72 (2011)
15. Amaral, R., Badia, R.M., Blanquer, I., Braga-Neto, R., Candela, L., Castelli, D., Flann, C., De Giovanni, R., Gray, W.A., Ourones, A., Lezzi, D., Pagano, P., Perez-Canhos, V., Quevedo, F., Rafanell, R., Rebello, V., Sousa-Baena, M.S., Torres, E.: Supporting biodiversity studies with the EUBrazilOpenBio Hybrid Data Infrastructure. Concurrency and Computation: Practice and Experience (2014). doi:10.1002/cpe.3238
16. Sanchez-Exposito, S., Martin, P., Ruiz, J.E., Verdes-Montenegro, L., Garrido, J., Sirvent, R., Ruiz Falco, A., Badia, R.M.: Web Services as building blocks for Science Gateways in Astrophysics. In: proceedings of Interntational workshop on Science Gateways, pp. 80–84 (2015). doi:10.1109/IWSG
17. Martin, P., Sanchez-Exposito, S., Ruiz Falco, A., Lorenzana, J., Ortega, E., Verdes-Montenegro, L., Garrido, J., Ruiz, J.E., Badia, R.M., Sirvent, R., Tejedor, E.: Business rules driven federated access to heterogeneous computing infrastructures with Web Services (in prep.)
18. Rogstad, D.H., Lockhart, I.A., Wright, M.C.H.: Aperture-synthesis observations of HI in the galaxy M83. Astrophys. J. **193**, 309 (1974). doi:10.1086/153164
19. Vogelaar, M.G.R., Terlouw, J.P.: The evolution of GIPSY, or the survival of an image processing system. In: Astronomical Data Analysis Software and Systems X, vol. 238, p. 358 (2001)
20. Ruiz, J.E., Garrido, J., Santander-Vela, J., Sánchez-Expósito, S., Verdes-Montenegro, L.: Astrotaverna - Building workflows with Virtual Observatory services. Astronomy and Computing I, 3 (2014)
21. Sengupta, C., Scott, T.C., Verdes Montenegro, L., Bosma, A., Verley, S., Vilchez, J.M., Durbala, A., Fernández Lorenzo, M., Espada, D., Yun, M.S., Athanassoula, E., Sulentic, J., Portas, A.: HI asymmetry in the isolated galaxy CIG 85 (UGC 1547). Astronomy and Astrophysics **546**, A95 (2012). doi:10.1051/0004-6361/201219948