CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2015; **27**:4361–4376
Published online 20 May 2015 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3538

SPECIAL ISSUE PAPER

Mobile application development exploiting science gateway technologies

Fabio Vitello[1,*,†], Eva Sciacca[1], Ugo Becciani[1], Alessandro Costa[1], Piero Massimino[1], Éva Takács[2] and Balázs Szakál[2]

[1]*Astrophysical Observatory of Catania, INAF, Italy*
[2]*4D SOFT Ltd, Hungary*

SUMMARY

Nowadays, collaborative applications are valuable tools for scientists to share their studies and experiences, for example, by interacting simultaneously with their data and outcomes giving feedback to other colleagues on how the data are processed. This paper presents a mobile application connected to a workflow-enabled framework to perform visualization and data analysis of large-scale, multi-dimensional datasets on distributed computing infrastructures. In particular, the usage of workflow-driven applications, through science gateway technologies, allows the scientist to share heavy data exploration tasks as workflows and the relative results in a transparent and user-friendly way. Copyright © 2015 John Wiley & Sons, Ltd.

Received 19 January 2015; Revised 23 March 2015; Accepted 15 April 2015

KEY WORDS: mobile application; workflow systems; science gateways; collaborative environments; astrophysics; large-scale datasets; scientific visualization; DCIs

1. INTRODUCTION

Mobile electronic devices such as tablets and smartphones are increasingly common. Individuals will frequently own a collection of these mobile devices. Yet, these devices are often resource limited: processing power is low, battery life is finite and storage space is constrained. These restrictions slow application execution, and hinder operability.

Distributed computing infrastructures (DCIs) offer an attractive alternative for resource-demanding applications. Currently, DCIs predominately serve computationally intensive scientific and enterprise applications. Several efforts combine DCIs and mobile devices. Chu and Humphrey extended OGSI.NET to mobile devices addressing the mobile devices' resource limitations and intermittent network connectivity [1]. Gonzalez-Castano *et al.* incorporated mobile devices into Condor as client front-ends for job submission and job querying to traditional supercomputer grids [2]. Please refer to [3] for a general survey on these topics. While addressing some of the particular concerns of mobile devices, none of these efforts embrace a large variety of DCIs apart from grid infrastructures taking advantage of the workflow system paradigm.

This paper presents the latest development and testing details and related issues of a mobile application, first reported in [4] and [5], which employs science gateway (SG) technologies [6] and workflow systems to connect to several DCIs.

*Correspondence to: Fabio Vitello, Astrophysical Observatory of Catania, INAF, Italy.
†E-mail: fabio.vitello@oact.inaf.it

The mobile application has been developed within the Scientific Gateway-based User Support project[‡] to support new access mechanisms to the science gateways' related technologies. The access to the science gateway through the application, instead of a web browser, allows to offer a user experience according to the mobile app design guidelines. In particular, it is based on the 'one click' approach to present the results in a simple and straightforward way.

The VisIVO Mobile application is connected to a Web Services-Parallel Grid Runtime and Developer Environment / grid and cloud User Support Environment (WS-PGRADE/gUSE) gateway [7] and accesses to VisIVO Server [8] tools, enabling execution of a comprehensive collection of modules for data exploration (including processing and visualization) of astrophysical datasets on DCIs. A number of customized workflows have been configured by default, for example, to allow local or remote upload of datasets and creation of scientific movies. These workflows are provided with specific user interfaces to enable easy parameter setting for the end users while hiding the complexity of the underlying system and infrastructures. The mobile application employs the same user accounts of the SG and provides a platform for astrophysical communities to share results, analysis experiences, and exploration of their datasets.

Even if the application has been developed connected and customized for the VisIVO Science Gateway, the integration with gUSE and Liferay also gives a unique opportunity to communities from other scientific disciplines for developing similar applications. The mobile application development allowed us to understand what kind of features of the science gateway framework had to be modified to be mobile compliant (such as the graph editor or the upload functionality to configure a concrete workflow) and to offer new functionalities such as the interactive visualization facility, which is a unique feature of the application.

A number of challenges has been faced during the development of the application; the main ones are related to the following activities: to design the application in order to access the functionalities in a simple and 'homogeneous' way in accordance with the ecosystem of the existing mobile applications (as shown in Section 4); to develop the authentication module because Liferay does not offer an API for authentication and it had to be implemented from scratch as discussed in Section 4.2; to investigate suitable integration modes with gUSE that we solved by using the remote API (described in Section 4.2); and, finally, to choose the proper testing procedure (Section 5).

The remainder of the paper is organized as follows: Section 2 discusses the related works. Section 3 is a brief introduction to the visualization tools. Section 4 describes the mobile application development and illustrates the main functionalities focusing on the implementation details, the employed technologies, and the underlying architecture. Section 5 highlights the issues related to test a mobile application and report the experiences of testing VisIVO Mobile. Finally, Section 6 presents future developments envisaged for VisIVO Mobile, and Section 7 outlines the conclusions.

2. RELATED WORKS

In [9], the authors describe their attempts to write grid clients for mobile devices, such as a Personal Digital Assistant (PDA), which have restrictive computational and storage facilities. Their experiences are based on an implementation of a mobile grid client for an existing web-based e-learning system. Also, in [10], it is presented a system infrastructure that allows local mobile devices to interact with the grid. Central to this infrastructure is a proxy with the ability of dual connectivity to transfer the request from the mobile device to the grid. This system infrastructure combines the mobility of mobile devices with the processing power of the grid.

We have previously developed a mobile application presented in [11] named SpaceMission[§]. It offers hands-on experience of astrophysical concepts using scientific simulations. It has been developed as a game that allows players to interactively fly into large-scale computer models of our cosmos. The player undertakes a virtual journey in a simulated model of our cosmos spanning several millions of earth years to discover galaxies. At any point during this journey, the player can click a snapshot using a virtual camera. Once all galaxies are found, cinematic movies can be

[‡]http://www.sci-bus.eu/.

[§]SpaceMission web page: http://spacemission.oact.inaf.it.

DOI: 10.1002/cpe

generated from the snapshots taken. The application is based on VisIVO tools for visual discovery through 3D views generated from astrophysical datasets.

Thanks to the usage of WS-PGRADE/gUSE framework, we have redesigned our application to be more general. It is possible to load and analyze any dataset, and the processing is not limited to grid infrastructures to perform heavy jobs but can be extended to any other DCIs handled by gUSE such as clusters or clouds.

3. VISUALIZATION TOOLS

The visualization tools employed by the mobile application are based on VisIVO [12], which is an integrated suite of tools and services for visual discovery within large-scale astrophysical datasets. VisIVO functionalities are exploited by the following tools:

- VisIVO Desktop [13], a stand-alone application for interactive visualizations running on PCs;
- VisIVO Server[¶], a grid-enabled high-performance data exploration and visualization command line toolkit;
- VisIVO Library [14] has been developed to port VisIVO Server on gLite middleware[‖] but can be installed on the most common DCIs; and
- VisIVO Science Gateway[**] [4], a workflow-enabled web portal wrapped around WS-PGRADE/gUSE framework providing visualization and data management services to the scientific community by means of an easy-to-use graphical environment for accessing the full functionalities of VisIVO Server.

Users of each of the aforementioned tools can obtain meaningful visualizations rapidly while preserving full and intuitive control of relevant visualization parameters. Any VisIVO tool accesses to the following main modules: VisIVO Importer, VisIVO Filters, and VisIVO Viewer.

VisIVO Importer converts user-supplied datasets into an internal representation called VisIVO Binary Table (or VBT). VisIVO Importer supports conversion from several popular formats such as ASCII and CSV or VOTables[††] without imposing, in principle, any limits on sizes or dimensionality. VisIVO Filters are a collection of data processing modules to assist the extraction of hidden properties of the dataset. These filters support a range of operations such as scalar distribution, mathematical operations, or the selection/extraction/merging of regions of interest. VisIVO Viewer is the visualization core component. It allows the rendering exploiting the Visualization ToolKit (VTK)[‡‡] (VTK) and Splotch [15], a ray-tracer visualization algorithm. It creates visualizations from multidimensional datasets, for example, by rendering points, volumes, and isosurfaces. There is support for customized lookup tables and visualizations using a variety of glyphs, such as cubes, spheres, or cones. VisIVO Viewer can be also used to produce images in a given range of values for azimuth, elevation, and zoom level that can be externally assembled to generate fly-through movies.

A typical astrophysical data exploration workflow involves the following tasks: first utilize VisIVO Importer to convert a user dataset into a VBT. Then, one or more VisIVO Filters are possibly applied to process these datasets and highlight the properties of interest, and finally VisIVO Viewer is invoked to produce the final visualization (Figure 1).

4. VISIVO MOBILE APPLICATION

The VisIVO Mobile application (Figure 1), available on the App Store[§§], allows iPad to exploit VisIVO Science Gateway functionalities to access large-scale astrophysical datasets residing on a server repository for analysis and visual discovery. The mobile application produces customized

[¶] VisIVO Server web page: http://sourceforge.net/projects/visivoserver.
[‖] gLite web page: http://glite.cern.ch.
[**] VisIVO Science Gateway web page: http://visivo.oact.inaf.it:8080.
[††] VOTable web page: http://www.ivoa.net/documents/VOTable.
[‡‡] VTK web page: http://www.vtk.org.
[§§] VisIVO Mobile App Store link: https://itunes.apple.com/it/app/visivo-mobile/id914448996?l=en&mt=8.

DOI: 10.1002/cpe

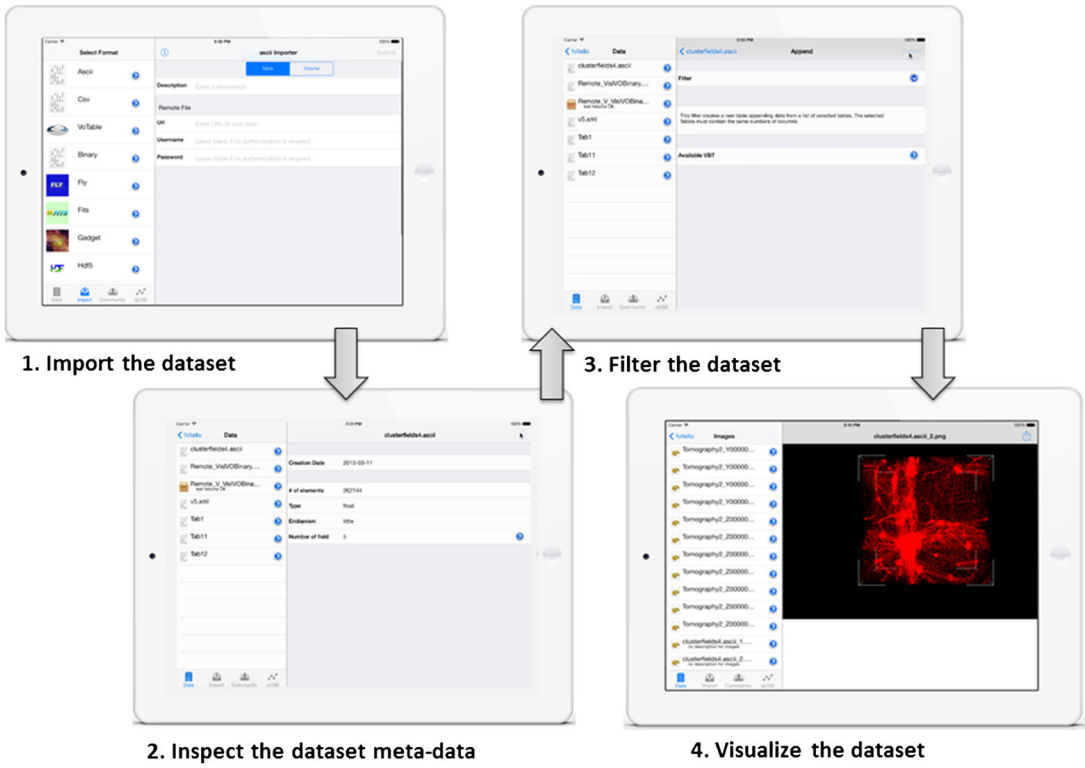

Figure 1. VisIVO Mobile screenshots on an iPad device: dataset remote importing, inspection of the relative metadata, dataset filtering, and visualization of the produced images and scientific movies.

visualizations (images or movies) on DCIs through interactive widgets submitting VisIVO-based workflows. For example, the importing interface shown in the left screenshot of Figure 1 employs a workflow to perform the remote upload of a dataset (see Section 4.1 for more details). Furthermore, it provides access to the WS-PGRADE/gUSE functionalities to create, configure, and submit new workflows.

The application allows the navigation through the imported datasets, produced images, and scientific movies, as shown in the screenshots in Figure 1, and notifies users when requested visualizations are available on the remote server for retrieving on their devices. Furthermore, it allows sharing of data, images, and movies via e-mail or by exploiting popular social networks. This provides a very handy way for scientific collaboration with familiar interaction environment such as Facebook. For more details on the available functionalities, a video demo is available on YouTube[¶¶].

The VisIVO application offers two modes of visualization: the VisIVO Viewer processing and the interactive viewer. The interactive viewer (Figure 2) allows an easy exploration of the dataset, giving a quick overview on the data, using simple typical touch-screen native gestures such as swipe, pinch to zoom, scroll, and rotate to explore a particular region of interest. In order to use the interactive visualizer, the dataset has to be downloaded to the mobile device and should fit the memory size to be processed. In this case, the original dataset size is reduced by running a workflow on DCI, which applies a Randomizer VisIVO Filter to extract a random subset (in terms of percentage) from the original table/volume; the workflow also converts the dataset to obtain the format required by the interactive viewer. This kind of visualization only applies to point rendering, while for more sophisticated visualization algorithms (e.g. volume rendering, isosurface, or tomography) or, for example, to apply different kind of palettes or glyphs, the VisIVO Viewer workflow processing on DCI is applied and static images or scientific movies are produced. We envisage to optimize and combine the two visualization approaches as discussed within the Section 6.

[¶¶]VisIVO Mobile video demo on YouTube: http://youtu.be/9c8yz9tCQnc.

DOI: 10.1002/cpe

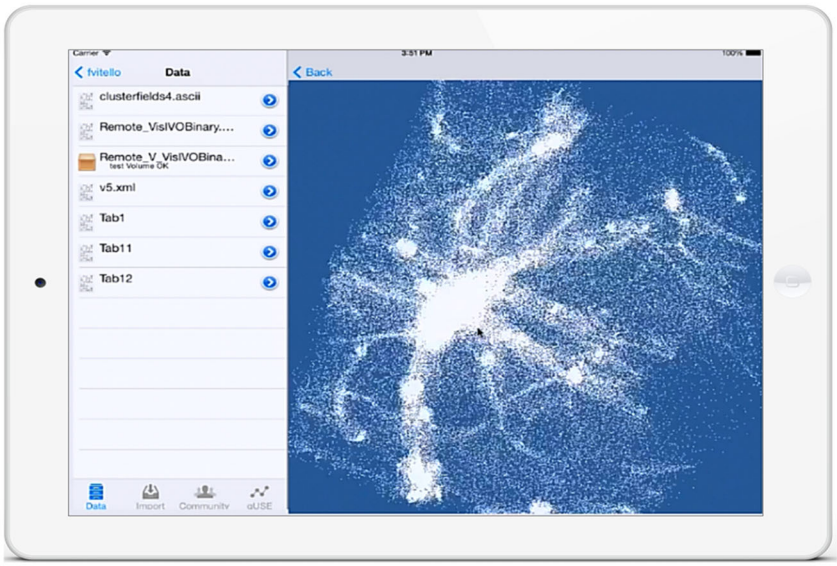

Figure 2. VisIVO Mobile screenshot on an iPad device: interactive visualization of a dataset.

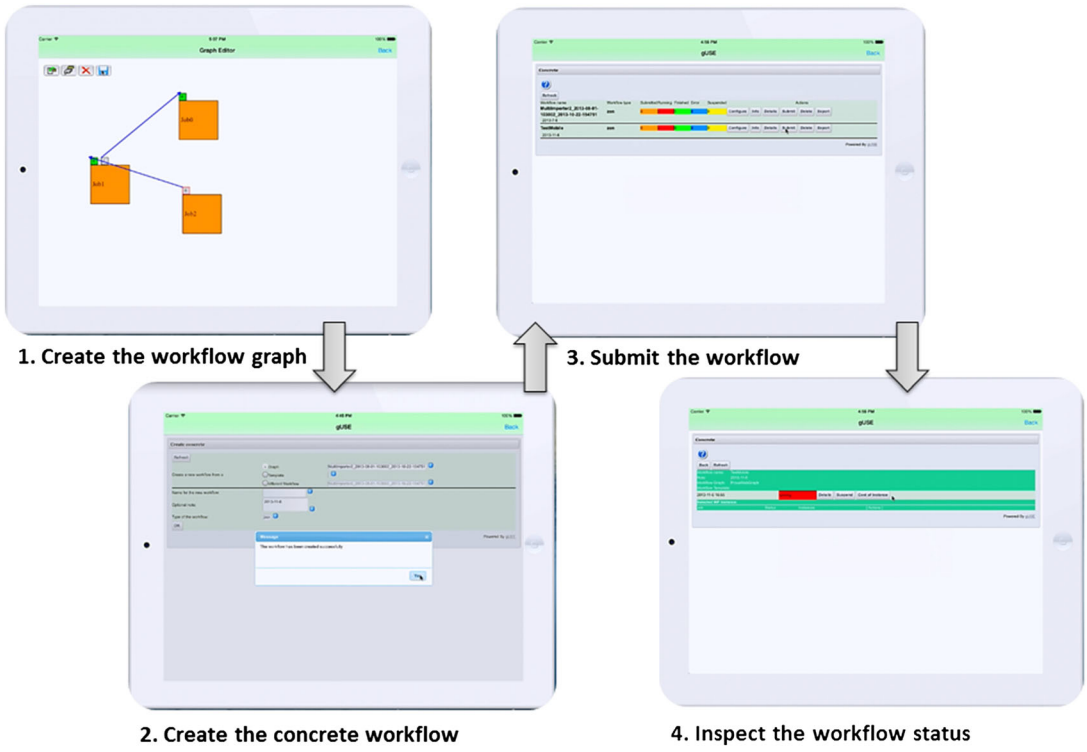

Figure 3. VisIVO Mobile screenshots on an iPad device: integrating WS-PGRADE/gUSE utilities to create, configure, and submit a workflow and to inspect the workflow running statuses and get the results.

Native WS-PGRADE/gUSE utilities have been integrated into the VisIVO Mobile to create, configure, and submit a workflow from scratch directly from the application. The first implemented utility was the graph editor employed to create a workflow skeleton. The new graph editor has been developed as a Web-based portlet, and mobile device users are able to use this feature because the only available graph editor within WS-PGRADE/gUSE could only be run as a separate Java Web Start application on desktop computers. The portlet is implemented on JavaServer Pages (JSP)

DOI: 10.1002/cpe

and makes use of HTML5 and Javascript employing the KineticJS library[lll] to create the workflow building blocks (e.g. boxes for jobs or arrows for channels) and interact with them.

Three other functionalities are integrated from the native WS-PGRADE/gUSE core modules for configuring, submitting, and inspecting a running status of a workflow. Please see Figure 3 for some representative screenshots of these utilities.

The overall UI has been designed to follow the basic guidelines[***], which are usually different from the ones related to desktop or Web applications. Because interaction with the finger is very different than designing for mouse, the UI controls (e.g. buttons) are designed larger for a finger with respect to desktop environments. The usage of virtual keyboard has been limited whenever possible employing for example ListBox or spin control to replace a text field because a virtual keyboard takes up the screen space and is usually typo-prone. Finally, only the primary functionalities (e.g. the ones related to input the mandatory parameters to carry out the VisIVO Importer, Filter, or Viewer operations) are displayed at the screen at a time while the secondary functionalities (e.g. the ones related to input the optional parameters) are hidden.

4.1. VisIVO workflows

One of the first deployed workflows for VisIVO Mobile was the Remote Importer. This workflow imports a remote dataset providing the relative URL and, if required, the user name and password for authentication. Depending upon the size of the datasets under consideration, remote uploads could last a long period. Therefore, VisIVO Mobile provides an asynchronous importing mode, and when the dataset has been properly imported, a follow-up e-mail will notify the user (in next releases of the application, push notifications are foreseen). The workflow employed for remote importing allows generation of significant information for metadata exploration, for example, statistics on data values, histogram calculation, and plotting or a sample extraction of uploaded datasets. Such metadata can be displayed through tapping on the data item.

Once the data file is imported, a sequence of simple actions is required to rapidly obtain meaningful visualizations. Typically, various filtering operations are performed, and VisIVO Mobile automatically displays all applicable filters allowing input of the relevant parameters. Finally, the interactive visualization tool or the VisIVO Viewer i employed for the dataset display.

VisIVO Mobile further allows users to generate scientific movies. These can be useful not only to scientists to present and communicate their research results, but also for dissemination purposes for example at museums and science centers for introducing complex scientific concepts to general public audiences. Users can create a *panoramic movie*, this movie is automatically generated by a camera motion path of 360° in azimuth and $+/-90°$ in elevation within the dataset's domain. *Customized movies* can be produced by intermediate snapshots specified as camera positions/orientations and the underlying workflow generates a movie with a camera path containing these specified positions/orientations. *Dynamic movies* can be created by interpolating several steps of a time evolution of a cosmological dataset. The user can browse a cosmological time evolution and choose two or more coherent datasets. The designed workflow will then produce the necessary number of intermediate VBTs by calculating particle positions and applying boundary conditions as necessary. This approach can be very useful, for example, in revealing galaxy formation or observing large-scale structures such as galaxy clusters. The creation of a movie represents a significant challenge for the underlying computational resources as often hundreds or thousands of high-quality images must be produced. For this reason, Parameter Sweep (PS) workflows [16] are employed. This is particularly relevant to the visualization-oriented workflows because they typically employ a large number of parameters that have to be varied within user-defined ranges and several hundreds to thousands of workflow executions might be necessary. PS workflows are executed through distributed parallel computations. The gUSE/WS-PGRADE infrastructure supports special ports (generator and collector) that enable the PS elaboration of a workflow: a single set of input files

[lll] KineticJS library web page: http://www.kineticjs.com/.

[***] iOS Human Interface Guidelines: https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG.

DOI: 10.1002/cpe

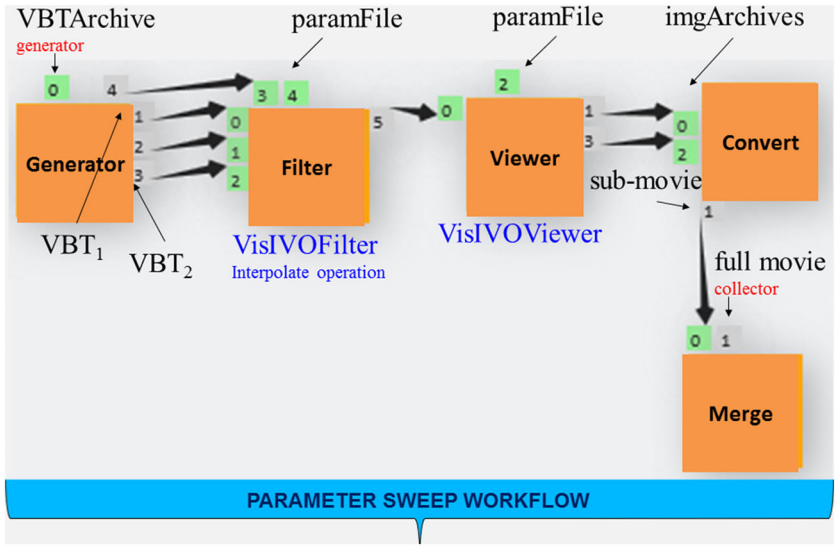

Figure 4. Dynamic Movie Workflow interpolates several steps of a time evolution of a cosmological dataset.

containing more than one element associated to a port – or several input ports having this feature – may trigger the proper number of submissions of the associated job. As an example, dynamic movies are generated with the workflow shown in Figure 4. Given n VBTs corresponding to n steps of a cosmological evolution, the employed workflow generates $n - 1$ sub-movies with different couples of VBTs: for each couple, the intermediates VBTs are calculated by interpolation, and all the images for the sub-movie are created for each interpolating VBT. The generation of these sub-movies is executed in parallel and is finally merged through a collector port as shown in the workflow depicted in Figure 4.

A number of challenging workflows have been prototyped to support highly specialized scientific communities mainly in astrophysics [4]. These workflows are supported in ER-flow[†††] project so that they can be stored into the SHIWA workflow repository together with related metadata, allowing investigation of their interoperability and dissemination across relevant communities through the SHIWA simulation platform [17]. VisIVO Mobile contains customized modules to submit these specialized workflows. For example, Figure 5 shows the interface to submit the *Muon Portal* workflow (described in [4] and [18]), which performs the visual inspection of cargo containers carrying high atomic number materials exploiting the deflection of muonic particles present in the cosmic radiations [19].

4.2. Architecture and implementation details

The VisIVO Science Gateway connected to the mobile application is based on the collaborative and community-oriented framework WS-PGRADE/gUSE. The gateway is fully integrated within the portal framework Liferay[‡‡‡], which is highly customizable thanks to the adoption of portlet technology, and compatible with modern Web applications.

An overview of the technology related to VisIVO Mobile is shown in Figure 6. The current version of the application is implemented in Objective-C optimized for the Apple iPad, and, in the near future, we are planning to port it to other popular smartphone devices. Even if a cross-platform solution (e.g. using HTML5 [20]) would have been more flexible to work seamlessly across different mobile platforms, we have chosen to build a native application because it can be highly optimized for speed and interaction in a way that will always surpass a third-party, remote solution such as HTML5 (as it has been confirmed on several surveys or blogs on the Web).

[†††]ER-flow web page: http://www.erflow.eu.
[‡‡‡]Liferay web page: http://www.liferay.com.

DOI: 10.1002/cpe

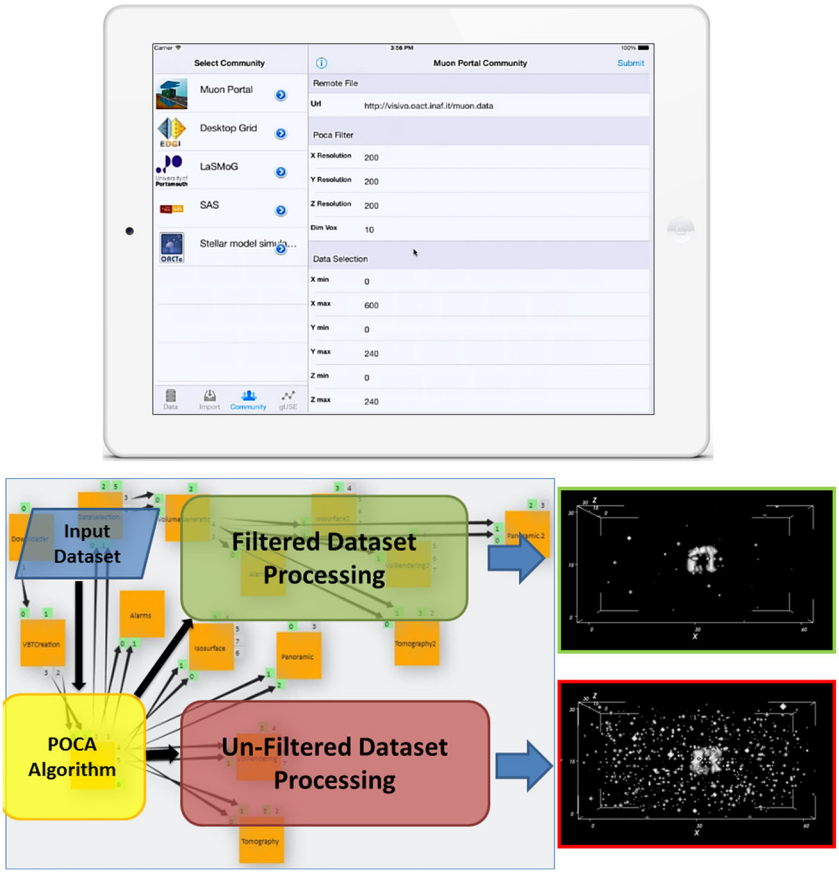

Figure 5. VisIVO Mobile screenshot of the Muon Portal interface and the submitted workflow with some resulting visualizations. POCA, Point of Closest Approach.

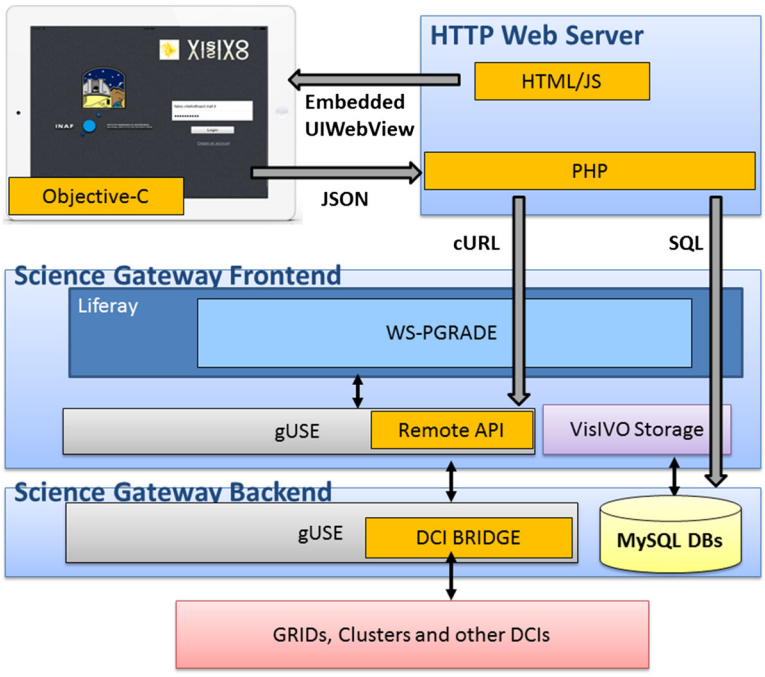

Figure 6. VisIVO Mobile framework architecture and employed technologies. DCI, distributed computing infrastructure; SQL, Structured Query Language; DBs, DataBases.

DOI: 10.1002/cpe

Listing 1: Remote API Usage Example in PHP

```
<?php
$url= "http://<UrlGatewayFrontend>:8080/
       wspgrade/RemoteServlet";
// remote submission of the workflow
$id=shell_exec("curl -k -F m=submit -F
       pass=$pwd -F wfdesc=@workflow.xml -
       F inputzip=@inputs.zip -F
       portmapping=@portmapping.txt -F
       certs=@certs.zip $url");
while(true){
// get the status of the submitted
       workflow
        $status=shell_exec("curl -k -F m=
               info -F ID=$id -F pass=$pwd
                $url");
        if(strcmp("error", $status)==0){
                break;
        }
        else if(strcmp("finished", $status
               )==0){
// download the output and log files of
       the submitted workflow when its
       status is finished
                shell_exec("curl -k -F m=
                       download -F ID=$id
                       -F pass=$pwd -o
                       output.zip $url");
        }
}
?>
```

The mobile application currently exploits the following two DCIs: the Cometa Consortium grid[§§§] distributed in seven sites of Sicily and an High Performance Computing (HPC) cluster hosted at the INAF Astrophysical Observatory of Catania.

The key component of WS-PGRADE/gUSE that allows the configuration and submission of the workflows to the DCIs through the gateway is the *remote API* [21]. This API interfaces to the core gUSE services without the WS-PGRADE user interface component hiding any details on the underlying technologies to the end users. Thus, running and managing scientific workflows are realized by cURL[¶¶¶]-based access. The API exposes usage of gUSE components through a simple Web service, resulting in wide adaptability by a diverse set of tools and programming languages.

The application connects to an HTTP web server to perform the databases queries and the gUSE Remote API calls implemented in Hypertext Preprocessor (PHP). The data are transferred using the JSON format, which is less verbose than XML. This is a very critical choice as data are transferred through the mobile network, which typically has a limited bandwidth. The remote API is implemented as one of the services of the front-end WS-PGRADE/gUSE components. The HTTP Web server also exposes some HTML/JavaScript Web pages embedded into the mobile application by means of the Objective-C UIWebView class (e.g. the mobile graph editor utility).

Listing 1 shows an example of the remote API usage from a PHP script. The WS-PGRADE workflow structure described into an XML file (`workflow.xml`) is submitted together with the input files and job binaries compressed into a ZIP archive (`inputs.zip`). The file `portmapping.txt` contains all the key–value pairs to identify the actual binaries and input files to the workflow jobs and input ports, respectively. The variable `$pwd` contains the single authentication password for the remote API usage.

[§§§]Cometa Consortium grid web page: http://www.consorzio-cometa.it.
[¶¶¶]cURL web page: http://curl.haxx.se/.

DOI: 10.1002/cpe

Listing 2 shows the Objective-C code required to connect to the HTTP Web server, which performs the gUSE remote API calls implemented in PHP. The parameter `url` identifies the location of the HTTP Web server, and `paramData` contains all the parameters needed for the execution of the remote API.

Listing 2: Objective-C code performing the calls to the HTTP web server

```
- (void)makePostRequest{
//JSON serialized object with all
       parameters needed for the execution
        of the Remote API
    NSData* jsonData = [
           NSJSONSerialization
           dataWithJSONObject:paramData
           options:kNilOptions error:nil];
// Create a JSON request to 'url' with the
        jsonData parameter
    _request = [[NSMutableURLRequest alloc
           ] init];
    [_request setURL:url];
    [_request setHTTPMethod:@"POST"];
    [_request setValue:@"application/json"
            forHTTPHeaderField:@"Content-
           Type"];
    [_request setValue:@"application/json"
            forHTTPHeaderField:@"Accept"];
    [_request setHTTPBody: jsonData];
// Make the connection with the created
       request
    [[NSURLConnection alloc]
           initWithRequest:_request
           delegate:self];
}
// Executed when a connection has finished
        loading successfully.
- (void)connectionDidFinishLoading:(
       NSURLConnection *)connection {
    //Put the connection results into a
           string
    NSString *results = [[NSString alloc]
           initWithData:receivedData
           encoding:NSASCIIStringEncoding
           ];
    //Convert the results string into a
           JSON Dictionary
    [receivedDictionary setDictionary:[[
           CJSONDeserializer deserializer]
            deserializeAsDictionary:[
           results dataUsingEncoding:
           NSUTF32BigEndianStringEncoding]
            error:nil]];
}
```

VisIVO Mobile requires the connection to an HTTP Web server and to the WS-PGRADE/gUSE gateway server using a mobile broadband network or a Wi-Fi connection. End users can log in with the same credentials as on the gateway, and the application provides the password coding in Secure Hash Algorithm (SHA) cryptography exploiting the built-in functionalities of the Liferay environment and querying the remote database to verify access credentials. In more detail, the current mobile authentication login process consists of the following steps: the user inserts the e-mail and password on the mobile application; the password is encrypted using the SHA1 algorithm and encoded in Base64; the encoded password and the e-mail are sent to the HTTP Web server using

DOI: 10.1002/cpe

a POST method in JSON format; the HTTP Web server checks the user e-mail and password connecting to the Liferay database and send back to the mobile application an authentication token; and, finally, the mobile interface lets the user access the application if the user credentials are valid, otherwise an error message is displayed.

As future work, we are planning to connect the application with an identity server, using, for example, the Lightweight Directory Access Protocol, to allow a federated authentication not only with the connected gateway (i.e. VisIVO SG) but also with the STARnet Gateway Federation [4], which is establishing a network of science gateways to benefit astrophysical communities sharing tools and services, data, repositories, workflows, and computing infrastructures. Furthermore, the security level of the connection will be increased by using HTTPS protocol and enhanced cryptographic algorithms such as SHA256.

The interactive visualization and data exploration are performed by means of the open-source VES/Kiwi framework[IIIII]. Kiwi provides easy-to-use building blocks for embedding advanced visualizations into Android or iOS applications. The Kiwi platform offers a collection of application-oriented C++ classes that tie together VTK's wide variety of supported file formats, readers, and filters, with the OpenGL ES 2.0 rendering capabilities of the VTK OpenGL ES Rendering Toolkit. It bundles together all the required rendering components, input/output (I/O) routines, and scene objects into a set of interfaces and offers a collection of 3D widgets designed for touch screens that allows users to manipulate the data and interact with the 3D scene.

5. MOBILE TESTING

The smart-appliances (phones and tablets) are rapidly becoming the primary method of interaction for businesses and also for scientific communities. As mobile applications become more and more important for users expecting higher-quality applications for mobile devices, application providers need to adapt and get ready to verify and evaluate mobile apps as part of their projects. Evaluating the quality of mobile applications is especially resource-intensive and time-consuming. Therefore, when it comes to testing, we need to find better and more cost-effective solutions to avoid any compromise on quality. Mobile test automation could help us to deliver higher-quality apps within certain budgets.

In many ways, mobile testing is more challenging than testing based on desktop and Web applications as mobile devices are less accessible and less open. There is a multitude of mobile devices with different screen sizes and hardware configurations, various operating system platforms, and frequent update of their versions (e.g. there are several differences between iOS 7.x and iOS 8.x). With each update, a new testing cycle is recommended to make sure no application functionality is impacted. Huge regression test suites need to be at hand.

Different mobile application types also pose a challenge on writing automated tests scripts to validate their functionalities. A mobile app can be a native app, a Web app, or a hybrid app, which has both contents. Testing of each such app type is different than another as their implementation is quite different from one another. As we see each app behavior from installation to functionality is different from one another, we understand that their testing and test coverage will also be different. In our case, the VisIVO Mobile app is hybrid, running on both iOS 7.x and iOS 8.x on iPad.

Additionally, during mobile automated testing, numerous test interfaces should be considered, particularly emulators, simulators, and different kind of real devices. Mobile emulators and simulators are important testing tools, and they enable us to verify general functionalities and to perform regular regression testing. The very character of emulators and simulators means testing is being conducted in an environment that is not real. However, the advantages of such tools are considerable, yet limited in scope, and should never be considered a substitute for physical devices. Using emulators and simulators in tandem with devices give us the best results. In our case, we used the iOS Simulator running in Xcode (development environment for iOS).

Summarizing the complexity of mobile test automation is even more daunting as the huge availability of free/commercial mobile test automation tools is increasing. We can state that this segment

[IIIII]VES web page: http://www.vtk.org/Wiki/VES.

DOI: 10.1002/cpe

Algorithm 1 Test case for the VisIVOImporter module

Require: a registered user having a the access to an importable input file on a remote machine **and** relative appropriate parameters for the input file

1: **if** the user tap on ***Import*** on the lower left panel **and** select ASCII on the left panel **and** set the URL of the input file on the right panel **then**
2: the user is ready to submit the job
3: **end if**
4: **if** the status is finished **then**
5: the user receives back an e-mail **and** the VBT will be available through the data management facility available tapping on ***Data*** on the lower left panel
6: **end if**

of software quality assurance is developing with astonishing rapidity. But, the tool developers are also facing the pace of mobile development technologies resulting in less reliable tools. Our experience is that no one tool can fulfill all the requirements. For VisIVO Mobile app, we tried three tools, namely SilkMobile, Ranorex, and MonkeyTalk.

On the one hand, as basic concepts, these three tools seem to be quite similar. They are all cross-platform testing tools that allow implementing automated functional tests for mobile that replicate end-user experience and ensure that the application works as expected. Basically, we can implement tests that replay more or less the end-user manual device interaction. Important features are recording and playback, object recognition methods, object repositories, parallel running of tests on multiple devices, gesture support imitating the user experience, including drag and drop, zoom, and scrolling, exporting the code in common programming languages, reporting, and continuous integration capabilities. In all the cases, the applications should be instrumented in order to be recognized by the tools. In the case of Objective-C, the source code is instrumented, in the case of Android, the bytecode. According to our experiences, the instrumentation process is straightforward, and the effort needed is less than 5% within the overall testing process. But, because of the nature of Objective-C and the closeness of iOS, the vendors often fail to produce error-free frameworks. This problem could result in severe errors influencing the testing process. With Android, we have not experienced instrumentation problems.

On the other hand, there are important differences between them in terms of interaction with internal emulators or external ones, object recognition methods, and the supported programming languages for scripting. For example, SilkMobile uses internal emulators, having the advantage that the tests can be recorded and performed on the base computer. But, this approach is error-prone, and we often faced resolution problems. Contrary to this approach, Ranorex does not have integrated internal simulators. The tests are recorded and performed on the built-in Xcode simulator having the advantage that the aforementioned screen resolution problem disappears. The disadvantage of this method is that the testing is performed through remote desktop connection. Another important difference between the tools is the scripting languages they apply. SilkMobile's test scripts can be imported in a variety of popular programming languages such as Java, Python, and C#, while Ranorex just in C#. Additionally, SilkMobile test suites can be integrated in popular build and continuous integration tools. Ranorex is designed to work better with Microsoft Visual Studio 2010, Team Foundation Server. Lastly, comparing the open-source tools and commercial ones, we notice that object repositories are common in commercial tools but sometimes are missing in open-source solutions.

In the following, we describe a VisIVO Mobile functional test implementation using the Ranorex tool. As for the device, the Xcode iPad simulator has been connected. The purpose of the functional test is to validate the Importer module on testable input files and input parameters. The test case pseudo-code is shown in Algorithm 1.

****SilkMobile web page: http://www.borland.com/Products/Software-Testing/Automated-Testing/Silk-Mobile.
††††Ranorex web page: http://www.ranorex.com/mobile-automation-testing.html.
‡‡‡‡MonkeyTalk web page: https://www.cloudmonkeymobile.com/monkeytalk.

DOI: 10.1002/cpe

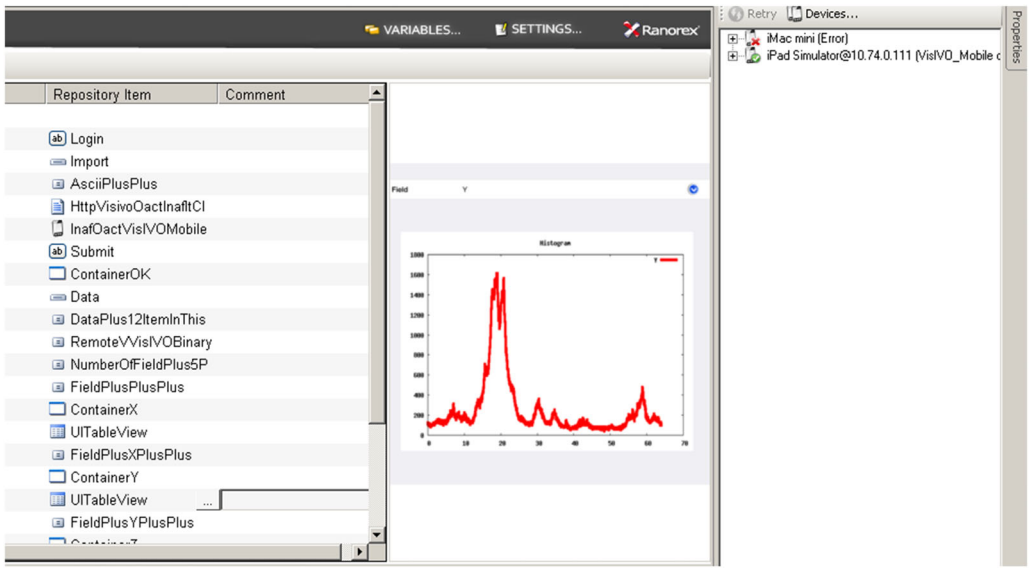

Figure 7. Ranorex editor employed to test the VisIVO Mobile.

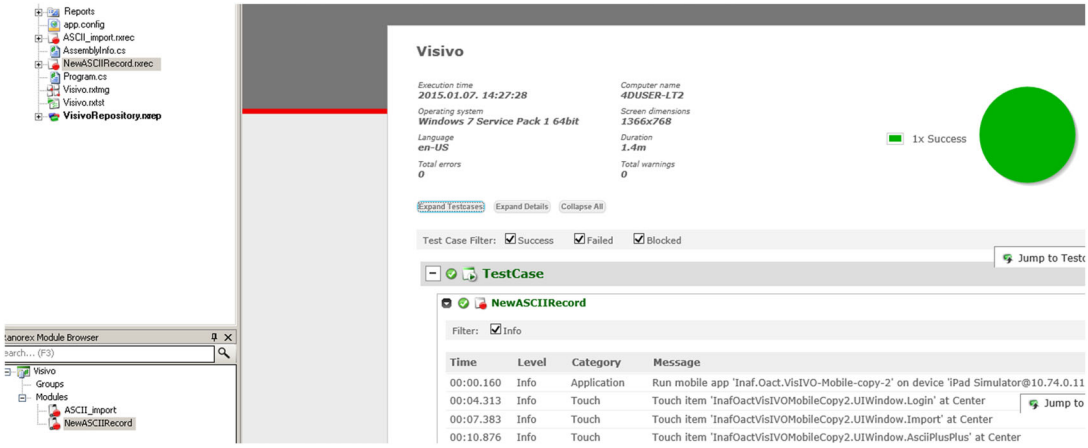

Figure 8. Ranorex report generated after the VisIVO Mobile test on the Importer module.

Firstly, the VisIVO Mobile app was prepared and instrumented for testing using a framework provided by Ranorex. Practically, a special instrumented build runs in Xcode iPad simulator. Ranorex connects to the simulator through Wi-Fi and remote desktop connection. Secondly, the test is recorded in the simulator. Then the recorded steps are refined; for example, verification commands have been added. The Ranorex tool is particularly good at recognizing native UI elements and generating maintainable XPath expressions for them using a dedicated path editor. Ranorex also offers a dedicated UI object repository to arrange and organize all UI elements needed during automation. Figure 7 shows the editor for refining the recorded script.

Lastly, the recorded script is replayed. A report is generated as shown in Figure 8.

To sum up, repetitive tasks such as regression testing are good candidates for automation. Regression testing is usually performed for every new release of a mobile app. With potentially multiple releases, it would be very helpful to have set up an automated testing system to execute a test on each and every release. Moreover, in the case of mobile apps it is not enough to test the functionality. Mobile users are sensitive to design. As for further research topics for mobile testing, test tool designers need to provide new ways for testing the design aspects of the applications.

DOI: 10.1002/cpe

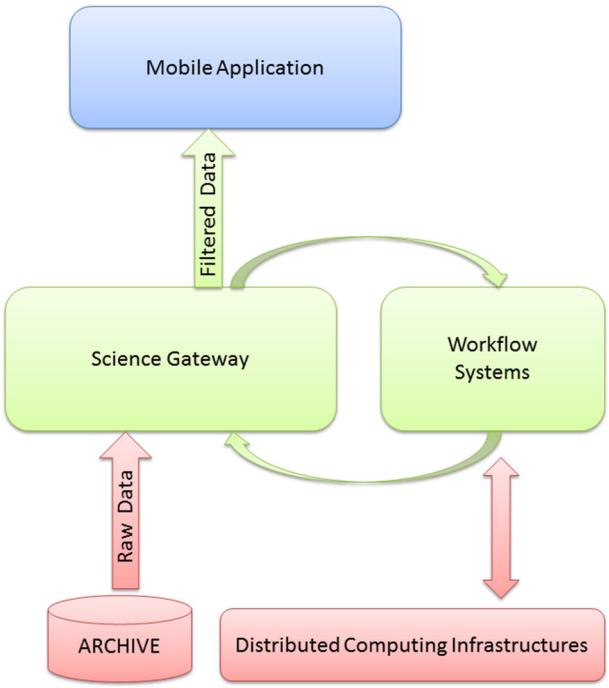

Figure 9. Envisaged architecture to connect the mobile application within a distributed and flexible framework to provide visualization of big data astronomical archives employing science gateways technologies.

6. FUTURE DEVELOPMENTS

As future developments, we envisage to connect the mobile application within a distributed and flexible framework to provide visualization of big data astronomical archives [22].

The framework consists of two main components: (1) remote services acting as interface between the main system backbone and the mobile client, dealing with requests from/to a data archive; and (2) the mobile application translates user interactions into remote server commands, showing the chosen archive visualizations efficiently handling big heterogeneous datasets, while providing a smooth and intuitive user experience.

Exploiting the Google Maps paradigm, the framework will pre-processes the bulk amount of data on a computing infrastructure, returning to the user with the specific processed data needed to render the current view of the archive on the screen. Each time a user interacts with the point of observation of the scene or changes a filtering method, the application should send the new parameters to a server that, in turn, will process the new subset of data needed.

In this case, science gateways technologies are employed to allow seamless integration of datasets, tools, and applications enabled for executing on different DCIs. The processes supported by gateways, organized as scientific workflows, are responsible for producing a reduced-size version of each content by applying efficient pre-processing operators such as mining/filtering algorithms. This provides the application with a single point of access exploiting the background computing infrastructures. The pre-processing workflows will have the task of enabling the remote services to perform operations that are central to the exploration of the underlying datasets appropriately orchestrating distributed resources, enhancing and ensuring application reusability. See Figure 9 for an exemplifying architecture of the proposed framework.

The proposed framework allows to bring data analysis and visualization into a single human-in-the-loop process. The navigation of 3D images of our galaxy with interactive touch-screen data

§§§§Google Maps web page: http://maps.google.com.

DOI: 10.1002/cpe

manipulation capabilities will provide powerful analytical, educational, and inspirational tools for the next generation of researchers. We are planning to develop the next-generation data analysis tools that will enable time-effective science for galactic star formation studies to interactively analyze a multi-dimensional galactic plane knowledge-base. Integration with data-mining and machine-learning technologies will enable developing new tools that incorporate data products, radiative transfer model libraries, and the astronomer's know-how into a set of supervised workflows with decision-making capabilities to carry out building of spectral energy distributions, distance estimate, and evolutionary classification of star-forming objects on the galactic plane.

7. CONCLUSIONS

This paper presents the development and testing details of VisIVO Mobile, an application tailored for the astronomy community connected to science gateway technologies to exploit several DCIs for data visualization and analysis. It gives the astronomy community a new tool to advance their research and opens doors to previously inaccessible opportunities from mobile devices. As the astronomy community uses it, we will evolve its design and implementation to optimize it for the particular workloads and access patterns observed. On the other hand, the modules to integrate native WS-PGRADE/gUSE utilities to create, configure, and submit workflows will be further enhanced and tested to make them more usable.

Finally, we hope that our overall framework and identified technologies will be exploited for the efficient and successful analysis and visualization of large scientific datasets on mobile devices in other fields of science and more in general to connect low-power local client applications to DCIs via science gateways.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Commission's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 283481 SCI-BUS (Scientific gateway Based User Support).

REFERENCES

1. Chu DC, Humphrey M. Mobile OGSI.NET: grid computing on mobile devices. *Proceedings. Fifth IEEE/ACM International Workshop on Grid Computing, 2004*, IEEE, Pittsburgh, PA, USA, 2004; 182–191.
2. González-Castaño FJ, Vales-Alonso J, Livny M, Costa-Montenegro E, Anido-Rifón L. Condor grid computing from mobile handheld devices. *ACM SIGMOBILE Mobile Computing and Communications Review* 2002; **6**(2):18–27.
3. Rodriguez JM, Zunino A, Campo M. Introducing mobile devices into grid systems: a survey. *International Journal of Web and Grid Services* 2011; **7**(1):1–40.
4. Becciani U, Sciacca E, Costa A, Massimino P, Pistagna C, Riggi S, Vitello F, Petta C, Bandieramonte M, Krokos M. Science gateway technologies for the astrophysics community. *Concurrency and Computation: Practice and Experience* 2014; **27**(2):306–327.
5. Vitello F, Sciacca E, Becciani U, Costa A, Massimino P, Takacs E. Developing a mobile application connected to a science gateway. *6th International Workshop on Science Gateways (IWSG), 2014*, IEEE, Dublin, Ireland, 2014; 12–17.
6. Kacsuk P. *Science Gateways for Distributed Computing Infrastructures*. Springer International Publishing: Switzerland, 2014.
7. 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* 2012; **10**(4):601–630.
8. Becciani U, Costa A, Comparato M, Caniglia G, Gheller C, Krokos M, Grillo A, Munzone R, Vitello F. VisIVO: new integrated services. *Astronomical Data Analysis Software and Systems XVIII* 2009; **411**:555–558.
9. Millard DE, Woukeu A, Tao F, Davis HC. Experiences with writing grid clients for mobile devices. *1st International ELeGI Conference on Advanced Technology for Enhanced Learning*, Vico Equense-Naples, Italy, March 2005; 1–8.
10. Guan T, Zaluska E, De Roure D. A grid service infrastructure for mobile devices. *First International Conference on Semantics, Knowledge and Grid, 2005. SKG'05*, IEEE, Beijing, China, 2005; 42–42.
11. Massimino P, Costa A, Becciani U, Krokos M, Bandieramonte M, Petta C, Pistagna C, Riggi S, Sciacca E, Vitello F. Learning astrophysics through mobile gaming. *Astronomical Society of the Pacific Conference Series* 2013; **475**:113.
12. Becciani U, Costa A, Antonuccio-Delogu V, Caniglia G, Comparato M, Gheller C, Jin Z, Krokos M, Massimino P. VisIVO–integrated tools and services for large-scale astrophysical visualization. *Publications of the Astronomical Society of the Pacific* 2010; **122**(887):119–130.

DOI: 10.1002/cpe

13. Comparato M, Becciani U, Costa A, Larsson B, Garilli B, Gheller C, Taylor J. Visualization, exploration, and data analysis of complex astrophysical data. *Publications of the Astronomical Society of the Pacific* 2007; **119**(858): 898–913.
14. Becciani U, Costa A, Ersotelos N, Krokos M, Massimino P, Petta C, Vitello F. VisIVO: a library and integrated tools for large astrophysical dataset exploration. *Astronomical Data Analysis Software and Systems XXI* 2012; **461**:505.
15. Dolag K, Reinecke M, Gheller C, Imboden S. Splotch: visualizing cosmological simulations. *New Journal of Physics* 2008; **10**(12):1–18.
16. Kacsuk P, Karoczkai K, Hermann G, Sipos G, Kovacs J. WS-PGRADE: supporting parameter sweep applications in workflows. *Third Workshop on Workflows in Support of Large-Scale Science, 2008. Works 2008*, IEEE, Austin, TX, USA, 2008; 1–10.
17. Kacsuk P, Terstyanszky G, Kiss T, Sipos G. Flexible workflow sharing and execution services for e-scientists. *EGU General Assembly Conference Abstracts* 2013; **15**.
18. 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. *5th International Workshop on Science Gateways, IWSG 2013.* CEUR Workshop Proceedings, Zurich, Switzerland, 2013; 1–8.
19. Bandieramonte M. Muon tomography: tracks reconstruction and visualization techniques. *Nuovo Cimento C - Colloquia and Communications in Physics* 2013; **44**(43).
20. Melamed T, Clayton B. A comparative evaluation of HTML5 as a pervasive media platform. *Mobile Computing, Applications, and Services*, Springer, San Diego, CA, USA, 2010; 307–325.
21. Balasko A, Farkas Z, Kacsuk P. Building science gateways by utilizing the generic WS-PGRADE/gUSE workflow system. *Computer Science* 2013; **14**(2):307–325.
22. Sciacca E, Pistagna C, Becciani U, Costa A, Massimino P, Riggi S, Vitello F, Bandieramonte M, Krokos M. Towards a big data exploration framework for astronomical archives. *Proceedings of the 2014 International Conference on High Performance Computing & Simulation (HPCS 2014)*, Bologna, Italy, 2014; 351–357.

DOI: 10.1002/cpe