Public Access Architecture Design of an FPGA-Hardware-based Nervous System Emulation Remote Lab

Gorka Epelde[1], Andoni Mujika[1], Peter Leškovský[1], Alessandro De Mauro[1]

[1] eHealth & Biomedical Applications, Vicomtech-IK4, Donostia - San Sebastian, Spain
{gepelde, amujika, pleskovsky, ademauro}@vicomtech.org

Abstract. This paper presents a public access architecture design of a remote lab developed for remote experimentation with an emulation of the nervous system of C. elegans nematode. The objective of this system is to emulate the neural system and the virtual embodiment of a C. elegans worm and to let the biologists' and neuroscientists' community remotely evaluate different neuronal models, and observe and analyse the behaviour of the virtual worm corresponding to the neuronal model being evaluated.
This paper first analyses related remote lab technologies which then is followed by the description of the adopted architecture design, selected cloud deployment option and remote user interface approach.

Keywords: Virtual Laboratories, Remote Laboratories, Nervous System Emulation, Neuro-computational Response Models on Field-Programmable Gate Arrays (FPGAs)

1 Introduction

The *Si elegans* project aims at providing a comprehensive artificial Caenorhabditis elegans (C. elegans) emulation system from which the principles of neural information processing, related to the behaviour of the C. elegans worm, can be derived.
The C. elegans hermaphrodite, a soil-dwelling worm, is well known worm regarding its cell composition. The morphology, arrangement and connectivity of each cell, including neurons, has been completely described [1]. Despite the extensive information on the worm's composition and behaviour [2], there is a lack of knowledge regarding the neural information processes that lead to the identified behaviours.
The *Si elegans* project is replicating the nervous system of the C. elegans on a highly parallel, modular, user-programmable, reconfigurable and scalable FPGA hardware architecture. It embodies the worm in a virtual environment for behavioural studies, taking the sensory-motor loop and realistic body physics into account. The resulting computational platform will be provided through an open-access web portal to the scientific community to test their neuronal models and different hypothesis.
This paper focuses on the architecture design defined to provide the scientific community with remote access to the neuro-computational research platform being developed.

2 Related Work Remote Labs Technology

A remote laboratory is commonly defined as an experiment which is conducted and controlled remotely through internet [3]. The main objective of *Si elegans* is to offer the scientists the possibility to remotely conduct behavioural experiments on an FPGA network-based hardware emulation system of the C. elegans worm. As such, *Si elegans* presents a remote laboratory designed for neuro-biological studies.

Given the benefit in terms of availability, administration and cost, remote labs have had a large proliferation during the last two decades. Two reference papers analyse and report on the state of art of remote laboratories [3, 4]. The research work in [5] takes these two reference papers, and provides an analysis from a design and development point of view for industrial electronics applications. The *Si elegans* project belongs to the industrial electronics remote laboratory category, considering that the core of the experiment is emulated in an FPGA network.

2.1 Existing Remote Lab Review

The analysis done in [5] concludes that most industrial electronics remote laboratories share a common architecture design. The main components of the conceptual architecture shared among many remote laboratories are: User interface, Web-server, Lab server, Instruments, Controller and Controlled Objects.

Our analysis of the remote labs has been mainly based on [3–5]. From the analysis we distinguished two different approximations: (i) toolkit initiatives for remote laboratories, (ii) FPGA specific remote laboratory implementations, and (iii) biological system simulation platforms. The identified remote labs have been examined by their software distribution approach (open source vs. commercial), last developer activity (for the open source projects) and basic functionalities, such as login and experiment booking.

Toolkits Initiatives for Remote Laboratories

iLab is a remote laboratory solution developed by the Massachusetts Institute of Technology [6]. It provides solutions for lab servers, clients, and a service broker (identified as web server in the generic architecture introduced above) to share different lab servers among clients in remote locations. Other toolkit initiatives provide gateways to integrate their solutions under the iLab service brokering. The solution has to be deployed using Microsoft based technologies (Windows Server, Structured Query Language (SQL) Server, .NET Framework). An Application Programming Interface (API) is provided in order to extend it freely.

iLab's software distribution approach is based on an open source license. The code is being distributed via subversion (SVN) code repository, manifesting frequent and recent updates.

WebLab-Deusto is a remote laboratory solution developed by the University of Deusto [7]. It offers solutions similar to iLab in regard to the lab servers, the clients, and the service broker. One of the benefits of WebLab-Deusto is the amount of extensions developed to make it interoperable with other remote laboratories (e.g. iLab) and to integrate it with learning management and authentication systems (i.e. LDAP, Oauth

2.0, and OpenID). Moreover, it offers seamless cross platform deployment (GNU/Linux, Mac OS X and Microsoft Windows systems).

For the client side, it allows the use of the Google Web Toolkit (GWT), JavaScript, Java, and Flash, although recommends the use of GWT to achieve maximised cross-browser interoperability. The lab server software is written in Python programming language. It differentiates between core and laboratory / experiment machine, allowing for a flexible distribution model (all in one machine, each of them in one machine) simply by changing related configuration files.

WebLab-Deusto's software distribution approach is based on an open source license. The code is being distributed via GitHub, also manifesting frequent updates.

Sahara Labs is a software suite developed by UTS Remote Labs that helps to enable remote access to computer controlled laboratories. It is designed to be a scalable and stable platform that enables the use and sharing of a variety of types of remote laboratories and maximises remote lab usage by implementing queuing and booking (or reservations) for users over a group of identical laboratories. The main language used for Sahara Labs development purposes is Java. The project is hosted at Sourceforge, and is being developed at GitHub. It has frequent updates.

Open Collaborative Environment for the Leverage of Online insTrumentation (**OCELOT**) is a complete solution framework for projects of remote collaborative interactions [8]. OCELOT software aims to provide a framework to quickly bring remote instrumentation solutions to the user. Its user interface is based on mixed reality and interactive multimedia.

OCELOT is open-source based. The implementation is based on the open-source implementation of OSGi enterprise server. The core components of a generic remote laboratory (User Interface, Web server, Lab Server and Controller as identified in the introduction) are launched in a centralised way on top of the OCELOT middleware.

The distribution is Git based, with logs showing recent activity only in internationalisation. The core development activity seems to have been abandoned. The last activity was registered on February 2012.

Laboratory Virtual Instrument Engineering Workbench (**LabVIEW**) is a system-design platform and development environment for a visual programming language by National Instruments (NI) [9]. LabVIEW allows for automation of processing and measuring equipment. LabVIEW provides integration of NI reconfigurable I/O (RIO) hardware targets through its FPGA module. In the introduced conceptual architecture the LabVIEW software provides user interface, web-server and lab-server component functionalities.

With regard to remote users, LabVIEW includes the support for remote panels, through which the developer can publish his new applications easily on a web-server. However, remote panels require a plug-in to run which is not available for mobile devices. Moreover, not all browsers support LabVIEW remote plugin, and the plugin approach breaks with the ongoing web development trends which use HTML5.

FPGA Specific Remote Laboratory Implementations

For the study of specific remote laboratory implementations we have focused on those tackling FPGA hardware configuration, learning and research. Following, two representative FPGA specific remote laboratory implementations are introduced.

Remote FPGA Lab is a web-based remote FPGA lab, developed by the National University of Ireland in Galway [10]. The main contribution of this development is the offer of novel web-based FPGA hardware design capability, and real-time control and visualization interface to a remote, always-on FPGA hardware implementation.

For the webserver, it uses the Django web-framework and Python programming languages. For the webpages, HTML and JavaScript is used. Webpage JavaScript supports the addition, placement and update of items on a predefined system block diagram, providing this way the definition of the console interface.

The Remote FPGA Lab is a research project having the design source not open, though licensable. The use of the application is open.

Distance Internet-Based Embedded System Experimental Laboratory (**DIESEL**) was developed by the University of Ulster [11]. The generic architecture consists of a gateway server (laboratory administrator) connected to the Internet and a number of experimental workstations connected to the server via a network-hub. This architecture is functionally composed of three interacting components: (i) a server based booking system, accessible through the web, which allows students to reserve a time slot on any available experimental workstation, (ii) a client application, which the end users install on their PCs to facilitate access to remote experimentation resources, and (iii) a server application, which runs on each remote workstation to facilitate the remote access process.

The DIESEL client–server approach uses a distributed software architecture developed using Microsoft .NET technology. The development is not active anymore.

Biological System Simulation Platforms

Traditionally most simulation platform models such as GENESIS [12], or NEURON [13] have been restricted to neurophysiological aspects. Recently the closed-loop interaction between neural circuits and biomechanics has been addressed. Animatlab software [14] allows to define a neural circuits and biomechanics interaction loop for any type of animal, while the OpenWorm project [15] targets specifically the C.elegans nematode. Most of the aforementioned biological simulation platforms are standalone applications that do not allow for remote multi-user experimentation. The only remarkable solution regarding the remote public access is the Geppetto simulation engine [16] being developed by the OpenWorm project, which offers a modular and web-based multi-scale simulation solution build over Java and OSGi technologies. The drawback of this solution regarding the *Si elegans* platform requirements is that it is targeted towards software-based simulation, and their interfaces and modular approach has not been designed for an implementation of a real time closed-loop between a hardware based neural emulation and a physically-based virtual environment calculations.

Beside the biological simulation platform discussed, the analysed generic remote lab toolkits are focussed on e-learning, whereas the *Si elegans* is focused on a specific research task. The Implementation of the *Si elegans* related research flows (e.g. definition of C.elegans behavioural experiments and the neuronal model and networks, and their conversion for the FPGA based emulation) do not follow a regular remote lab experimentation. In consequence, the actual development is not grounded on any of the analysed remote lab technology as such, but will individually reuse some of the open-source toolkit modules available (e.g. booking system).

3 Architectural Design of the *Si elegans* platform

Based on the requirement to provide public access to the scientific community, a cloud-based architecture has been defined for the *Si elegans* platform. The aim of defining a cloud-based architecture is to allow for parallel activity of scientists in: (i) defining new neuronal models and networks, (ii) defining new behavioural experiments, (iii) booking for emulation of any previously defined behavioural experiment run on a defined neuronal model and network, (iv) analysing existing results from previous emulations, and (v) collaborating at distance and contributing to research by sharing and discussing new models and results.

3.1 *Si elegans* Platform

The final design of the *Si elegans* architecture divides the solution into two main blocks: (i) public access services (provided through web technologies relying on cloud services), and (ii) hardware based C. elegans worm emulation with exclusive access. The second part, takes into consideration that the FPGA hardware implementing the nervous system of the worm, although being versatile and reconfigurable, is dedicated only to a single worm at a time. Functionalities that do not require a direct access to the emulation system are provided through the public access part (e.g. definition of new experiments and models, revision of results and the collaboration with other researchers). This way, in case an emulation is already running or a downtime on the hardware based emulation system occurs, the scientists can continue using the public access functionalities. Thus they can continue with their research. Figure 1 depicts the separation of the public access and the hardware access logic layers.

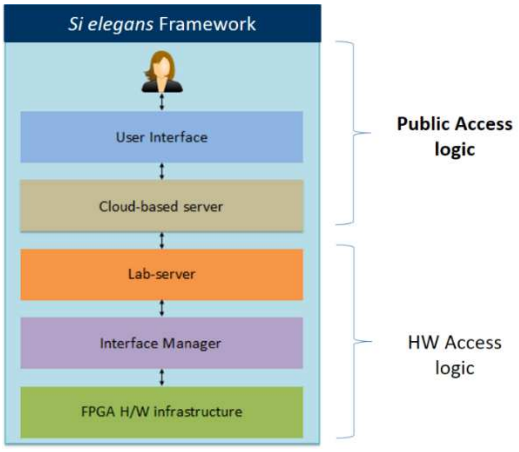

Fig. 1. Public Access logic identification over the *Si elegans* top level diagram.

The blocks defined for the *Si elegans* framework have been selected based on the main components (User interface, Web-server, Lab server, Instruments, Controller and

Controlled Objects) identified by Tawfik et al. at [5] as part of the conceptual architecture shared among industrial applications remote laboratories. In our design, the Web-server is referred to as Cloud-based server, in order to emphasize its cloud deployment nature. The Interface Manager block corresponds to the Controller component in Tawfik et al. paper, being responsible for controlling the FPGA-based neuron network, which in this case would correspond to the Controlled object in Tawfik et al. paper.

Figure 2 represents the system level components identified for the development of the public access logic.

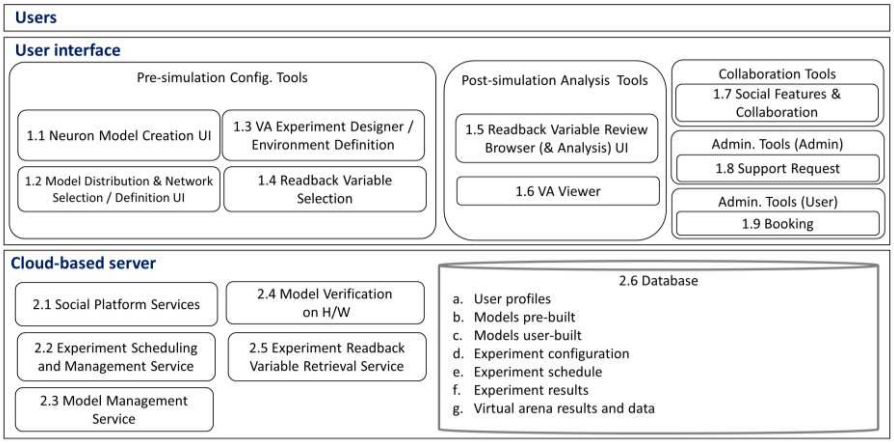

Fig. 2. System level components identified for the public access logic.

User interface components represent the visual web components that will be shown to the user, while the cloud-based server components pertain to the services that support those UI functionalities and launch HW Access logic actions. Virtual arena (VA) refers to the visualisation of the virtual 3D worm and the environment. Readback relates to monitoring of the neuron model parameters for debugging purposes. For the development of the Cloud-based server logic and to implement the required research flows we are using the Django web-framework and Python programming languages. In addition, we are considering both, the WebLab-Deusto and the Remote FPGA lab solutions for the reuse of specific blocks (e.g. booking and scheduling module).

Figure 3 represents the system level components identified for the development of the hardware public access logic.

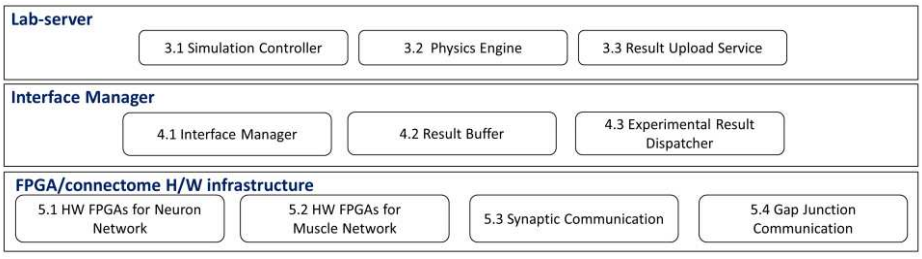

Fig. 3. System level components identified for the hardware access logic.

The HW access logic block is composed of the lab server, interface manager and the FPGA HW infrastructure. The lab server is the responsible for: (i) controlling at a high level the emulation (start, stop, and restore the system from a breakpoint), (ii) hosting the physics engine (for the physics calculation of the virtual worm and its environment), and (iii) uploading results to the cloud. The interface manager is responsible for initialising and managing the FPGA HW infrastructure (in addition to buffering and dispatching neuron variables readback information). Finally the FPGA HW infrastructure is composed of 302 FPGAs dedicated to the simulation of neurons (one FPGA per each C.elegans worm neuron), additional FPGAs used to accelerate muscle response calculations, and hardware with logic controller for the optical parallel communication between neurons.

Information on the implementation of the platform can be found for the FPGA HW infrastructure at [17], for the physics-based simulation of the virtual environment at [18], for neuron modelling to FPGA conversion process at [19], and for the optical interconnection technology being developed at [20].

3.2 Remote User Interaction with the Virtual Environment

An additional challenge of the public access design is the availability of remote interaction with the virtual 3D worm and its environment (whether in real-time or predefined). User interfaces tackling other *Si elegans* platform functionalities (e.g. defining new neuronal models and networks, collaboration with other users) are not part of this analysis since they are foreseen to be provided through regular web development technologies.

Decision must be taken on where and how the experiments are defined and the results of the physical emulation (the worm's behaviour) are represented and accessed.

Tawfik et al. [5] provide a good insight into the user interface technologies used to provide interaction with industrial electronic applications related to remote labs. They have identified the use of the following technologies: Desktop Sharing, Common Gateway Interface (CGI), ActiveX and Java Applets, Rich Internet Applications (RIAs), and Asynchronous JavaScript and XML (AJAX).

For the remote access to the virtual 3D worm of the *Si elegans* project, two possible scenarios have been considered: synchronous access (at the time of emulation), and asynchronous (no direct interaction while emulation is running) experiment definition and results analysis.

We have considered to offer the synchronous interaction through end-user tools like Virtual Networking Computing (VNC) and mixed web approaches [21]. The asynchronous interaction was considered to be provided through web 3D technologies, i.e. the experiment will be predefined, then carried out in the FPGA network and then visualized in a 3D environment with no real-time interaction.

We finally promote the asynchronous approach due to the following issues:

- The necessity of an exclusive access to the FPGA hardware in the *Si elegans* project limits the availability of the emulation system for multiple researchers at a time.

- Large communication latencies of the web limit the synchronous interaction with the 3D Virtual Arena. Basic advice suggesting acceptable response times of interactive applications [22, 23] of 100ms latency (considered to be the limit for users to experience direct object manipulation in an UI), together with the necessity of an immediate response in sensory/neural experiments make the synchronous access in overseas internet communication prohibitive.

We are developing the experiment configurator as a web 3D application, providing a 3D GUI, and the simulation results renderer as a distinct web 3D application, accessible once the hardware emulation engine and the physics engine finish computing the experiment results. The planned remote UI technologies are based on HTML5, AJAX and WebGL technologies.

3.3 Cloud Deployment

In the *Si elegans* project a cloud based approach has been planned, given: (i) most existing close-loop (between neural circuitry and biomechanical simulation) biological simulation platforms are standalone and to allow remote users to emulate their experiments and collaborate with other scientists, and (ii) ensure availability of as many platform functionalities as possible and to reduce maintenance tasks as possible.

The aim of this subsection is to introduce the different cloud technology service levels, and to decide the best entry point (selected service level) for the *Si elegans* public access platform development.

Cloud services are designed to provide services at different architecture layers [24], dividing it to hardware, infrastructure, platform and application services. Starting at the infrastructure level, each layer can be implemented as a service for the next, higher level layer. Gray [25] considers the following three service based business models:

- Infrastructure as a Service (IaaS): IaaS provides hardware and software infrastructure resources over the Internet. These resources are offered as on-demand services and are usually based on virtual machines assignment.
- Platform as a Service (PaaS): PaaS provides platform level resources on-demand. Making use of services at this level, the application developer does not need to interact at the virtual machine level but rather interacts with the higher level APIs provided by the cloud service.
- Software as a Service (SaaS): SaaS offers end-users applications as on-demand services through the Internet.

In our cloud-based approach, the PaaS technologies have been selected as an entry point. Nevertheless, within *Si elegans* we will develop a SaaS level service to provide public access to the hardware emulation system. The selection of PaaS entry level instead of the IaaS level has been motivated by the effortless system maintenance, maximising the system's uptime, the moderate maintenance costs and the facilitated scalability of the final platform. Event though, the project might consider moving a VHDL compiler solution (software to compile FPGA design file to FPGA programming binaries), or moving the physics engine (visual 3D worms physics simulation) to the cloud

to take advantage of the reduced maintenance tasks. To move these modules to the cloud IaaS access would be required. Therefore, PaaS service provider selection focusses on solutions that leave the door open to future access at a lower level (IaaS) in order to integrate more specific services (e.g. Amazon Beanstalk, Google Managed VMs).

4 Conclusions

In this paper we present the analysis done and the decisions taken with regard to the public access architecture design of an FPGA-hardware-based nervous system emulation remote lab.

First, the main toolkits used for remote labs and existing FPGA based remote lab development experiences have been analysed. Furthermore, following the main components identified by Tawfik et al [5] for remote laboratories for industrial electronics applications, we have defined our high level platform architecture. Based on the high level platform architecture, the components required for the public access logic have been identified.

Following, the approach for the remote access to the experiment configuration and results visualisation user interface have been analysed. The asynchronous experiment definition and results revision approach have been selected. The planned remote UI technologies will be based in HTML5, AJAX, and WebGL technologies.

For the deployment of our solution on the cloud, a PaaS approach has been selected, but with a possibility for IaaS level access to allow integrating services (e.g. VHDL compiler) that require a direct deployment on the virtual machine.

Actual work is focussed on the development of the system level components for the public access logic and the definition of the communication protocol between the public access logic and the hardware access logic.

Acknowledgments

This work was partially funded by the EU 7th Framework Programme under grant FP7-601215 (*Si elegans*).

References

1. White, J.G., Southgate, E., Thomson, J.N., Brenner, S.: The Structure of the Nervous System of the Nematode Caenorhabditis elegans. Philos. Trans. R. Soc. Lond. B Biol. Sci. 314, 1–340 (1986).
2. Hart, A: Behavior. The C. elegans Research Community, WormBook (2006).
3. Gomes, L., Bogosyan, S.: Current Trends in Remote Laboratories. Ind. Electron. IEEE Trans. On. 56, 4744–4756 (2009).
4. Gravier, C., Fayolle, J., Bayard, B., Ates, M., Lardon, J.: State of the Art About Remote Laboratories Paradigms–Foundations of Ongoing Mutations. Int. J. Online Eng. 4, 1-9 (2008).

5. Tawfik, M., Sancristobal, E., Martin, S., Diaz, G., Castro, M.: State-of-the-art remote laboratories for industrial electronics applications. In: Technologies Applied to Electronics Teaching 2012. pp 359–364. IEEE Press, New York (2012)
6. Massachusetts institute of technology: iLab Project, http://ilab.mit.edu/wiki.
7. University of Deusto: WebLab - Deusto, http://www.weblab.deusto.es/website/.
8. Ocelot : Opensource software, http://ocelot.telecom-st-etienne.fr/.
9. National Instruments: LabVIEW, http://www.ni.com/labview/.
10. Morgan, F., Cawley, S., Newell, D.: Remote FPGA Lab for Enhancing Learning of Digital Systems. ACM Trans Reconfigurable Technol Syst. 5, 1–13 (2012).
11. Callaghan, M.J., Harkin, J., McColgan, E., McGinnity, T.M., Maguire, L.P.: Client–server architecture for collaborative remote experimentation. J. Netw. Comput. Appl. 30, 1295–1308 (2007).
12. Genesis: The GENESIS Simulator, http://www.genesis-sim.org/GENESIS/.
13. NEURON: NEURON Simulator, http://www.neuron.yale.edu/neuron/.
14. Cofer, D., Cymbalyuk, G., Reid, J., Zhu, Y., Heitler, W.J., Edwards, D.H.: AnimatLab: a 3D graphics environment for neuromechanical simulations. J. Neurosci. Methods. 187, 280–288 (2010).
15. OpenWorm: The OpenWorm project, http://www.openworm.org/.
16. OpenWorm: Geppetto Simulation Engine, http://www.geppetto.org/.
17. Machado, P., Wade, J., McGinnity, T.M.: Si elegans: FPGA hardware emulation of C. elegans nematode nervous system. In: 6th World Congress on Nature and Biologically Inspired Computing. pp. 65–71. IEEE Press, New York (2014).
18. Mujika, A., de Mauro, A., Robin, G., Epelde, G., Oyarzun, D.: A Physically-based Simulation of a Caenorhabditis elegans. In: 22nd International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision - WSCG 2014, Vaclav Skala - Union Agency, Plzen (2014).
19. Krewer, F., Coffey, A., Callaly, F., Morgan, F.: Neuron Models in FPGA Hardware A Route from High Level Descriptions to Hardware Implementations. In: 2nd International Congress on Neurotechnology, Electronics and Informatics. pp. 177-183, SCITEPRESS, Lisbon (2014).
20. Petrushin, A., Ferrara, L., Liberale, C., Blau, A.: Towards an Electro-optical Emulation of the C. elegans Connectome. In: 2nd International Congress on Neurotechnology, Electronics and Informatics. pp. 184-188, SCITEPRESS, Lisbon (2014).
21. webtoolkit: Wt, C++ Web Toolkit, http://www.webtoolkit.eu/wt.
22. Bochicchio, M.A., Longo, A.: Hands-On Remote Labs: Collaborative Web Laboratories as a Case Study for IT Engineering Classes. IEEE Trans. Learn. Technol. 2, 320–330 (2009).
23. Card, S.K., Robertson, G.G., Mackinlay, J.D.: The Information Visualizer, an Information Workspace. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 181–186. ACM, New York, NY, USA (1991).
24. Zhang, Q., Cheng, L, Boutaba, R.: Cloud computing: state-of-the-art and research challenges. J. Internet Serv. Appl. 1, 7–18 (2010).
25. Gray, M.: Cloud Computing: Demystifying IaaS, PaaS and SaaS, http://www.zdnet.com/news/cloud-computing-demystifying-iaas-paas-and-saas/.