Migrating Healthcare Applications to the Cloud through Containerization and Service Brokering

François Andry, Richard Ridolfo and John Huffman
Philips Healthcare, 4100 E. Third Avenue, Suite 101, Foster City, CA 94404, U.S.A.

Keywords: Cloud, IaaS, PaaS, SaaS, Containerization, Security, Interoperability, Data Privacy, Healthcare, Wellness Applications, Big Data, Micro Services, IHE, High Availability, Performance, Elasticity, Development, Monitoring, Support.

Abstract: New business models and technologies offer unique opportunities of combining patient demographics and clinical data with general consumer data. We are building a digital health platform using a new paradigm based on an open platform as a service (PaaS) that delivers data and analytics across a wide variety of cloud computing topologies. This new architecture gives us the ability to integrate devices, data sources and services very quickly to create, refactor, migrate, deploy and maintain scalable, secure, high quality healthcare and wellness applications while reducing the total cost of ownership.

1 INTRODUCTION

New business models and emerging technologies offer unique opportunities for existing healthcare solutions vendors to undertake strategic reengineering of their technology and infrastructure stack, development process, deployment and support models. This is even more relevant for large organizations where years of organic growth and acquisitions and often have led to IT, technology and information systems silos.

These new models are based on cloud computing which offers on-demand access to a shared pool of configurable and elastic computing resources (networks, servers, storage, services).

The goal is to reduce IT expenses and operating costs by purchasing processing, bandwidth and data storage resources as needed. This is particularly critical in healthcare where competition, new delivery models and commoditization is forcing solution vendors to envision new ways to cut cost (Armbrust et al. 2010) and bring new applications and products to the market faster.

Organizations that are building their own cloud infrastructure from scratch or rely uniquely only on an infrastructure as a service (IaaS) from a provider, risk spending valuable resources and time building a specialized platform instead of focusing on their core business. On the other hand, organizations who adopt a turnkey proprietary cloud stack will lack flexibility and may end up locked into a specific technology or vendor.

Instead of designing the cloud architecture from the bottom up or the top down, a better strategy is to design from the inside out. By starting with the platform as a service (PaaS) as the central critical layer and creating ways to use various IaaS models and offerings in generic ways, it is possible to create a flexible and efficient lifecycle for the services and applications running on the platform.

In fact, PaaS that are built on top of IaaS layers (He et al. 2013) are now becoming the central layer for building cloud based applications (Vaquero et al. 2011).

In healthcare, this new cloud model facilitates the rapid creation and migration of existing applications towards better user engagement, increasing collaboration between care givers and improving the lives of patients, while reducing the total cost of ownership (TCO).

2 PLATFORM FOUNDATION

The foundation of our digital healthcare platform is built on Cloud Foundry, a new generation PaaS architecture. At the core of this platform is an elastic runtime self-service application execution component, coupled with an automation engine for application deployment and lifecycle management.

Andry F., Ridolfo R. and Huffman J..
Migrating Healthcare Applications to the Cloud through Containerization and Service Brokering.
DOI: 10.5220/0005249601640171
In *Proceedings of the International Conference on Health Informatics* (HEALTHINF-2015), pages 164-171
ISBN: 978-989-758-068-0

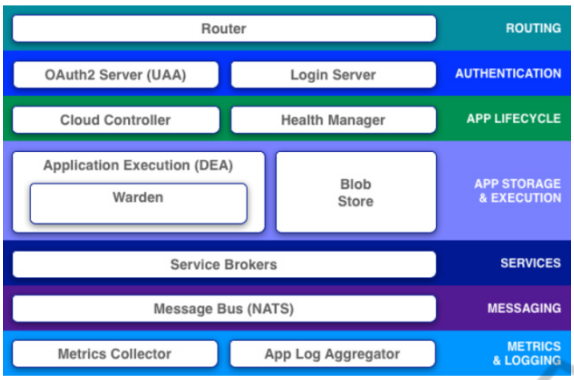

Figure 1: Cloud Foundry PaaS Components.

The router is responsible for dispatching incoming traffic (e.g., client applications requests) to cloud controller to a running application contained in a droplet execution engine (DEA) node.

The authentication layer provides identity management via OAuth2 and a login server component.

The cloud controller and the health manager manage the lifecycles of the applications hosted in the platform.

The Cloud Foundry unit of execution is called a droplet. It is pre-built, pre-configured, stored in the blob store, and dynamically deployed inside a Cloud Foundry DEA node as needed.

The message bus uses network address translation (NAT), queueing and a publish/subscribe mechanism for internal component communication and outbound traffic management.

The service brokers are components that provide back-end service instances and bind these services to an application at runtime.

Metrics collector and log aggregator components are used to collect events for developers and operators of the platform.

The main characteristics of this platform are:

- Application containerization
- Optimized application scaling
- Application to service brokering
- Abstraction of IaaS
- Excellent application lifecycle management
- Automatic middleware stack and operating system configuration
- Advanced application monitoring

2.1 Containerization

The principle of containerization is to ensure that application instances run in isolation without interference from other tenant applications while retaining full access to their assigned and dedicated share of resources from the IaaS layer. In this platform, application instances live inside a warden container, which provides an API for managing the creation, configuration, usage and destruction of these isolated environments.

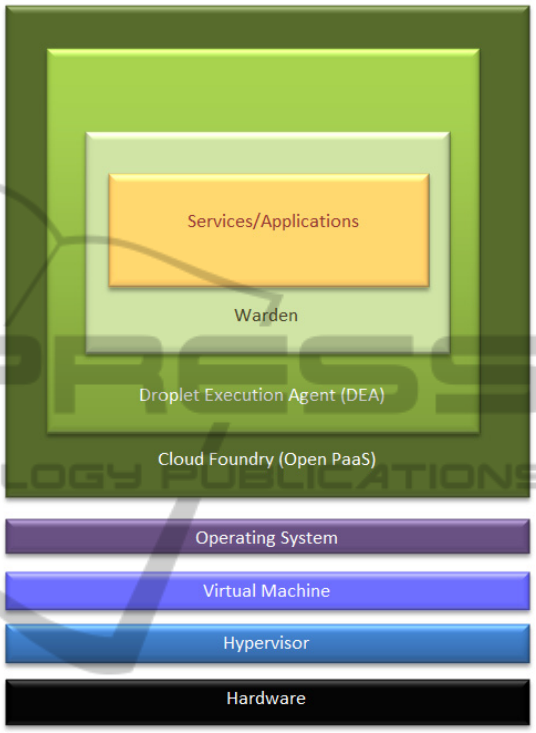

Figure 2: Containerization in Cloud Foundry.

Isolation is achieved by associating name spaces to the underlying operating system kernel resources. As a result, each container has its own network, Process ID and mount namespaces.

Each container is assigned a network interface (managed by the NAT component), offering fine-grained network traffic network management at the container level. In addition to this, containers also receive a private root file system.

Control groups help manage resources and provide a way to precisely control memory, CPU, disk and network access for each container.

2.2 Elasticity and Scalability

Above all, Cloud Foundry manages elasticity extremely well. In fact, it is considered by many that “elasticity, is the true golden nugget of cloud computing, and brings to the IT infrastructure what Henry Ford has brought to the automotive industry with assembly lines and mass production:

affordability and substantial improvements on time to market" (Owens, 2010).

Application software needs to scale down as rapidly as it scales up, which is a new requirement (Armbrust et al. 2010).

Applications deployed on the open PaaS can be scaled up and down extremely rapidly without any loss of transaction or data, which is particularly critical when dealing with a healthcare application.

The platform has been designed with specific new cloud application design principles in mind, such as those that can be found at http://12factor.net. One of these principles states that processes inside a container are disposable and can be started or stopped at any time. As a result, applications must be stateless so no local data is lost. The platform shuts down processes gracefully by refusing any new request while completing the current transaction, thereby making the corresponding operation idempotent.

Cloud Foundry scales up and down by provisioning or terminating application instances extremely quickly inside a set of DEA nodes, across availability zones.

In addition, the platform has four mechanisms to ensure a high level of availability:

- Automatic reboot of a container when an application fails
- Automatic reboot of the platform component in a new virtual machine (VM)
- Built-in VM monitoring to mitigate operating system (OS) failures
- Spreading applications across availability zones to mitigate geographic failures

2.3 Service Brokering

In this architecture, backing services (e.g., databases, caching systems, other data services (e.g., Amazon S3), messaging/queueing systems, SMTP services, various external APIs (Google Maps, terminology services, healthcare registry services) are just attached resources. For example, there is a distinction between a local digital imaging and communications in medicine (DICOM) local image store and a remote, 3rd party DICOM picture archiving and communication system (PACS) service hosted in the cloud (Bastião Silva et al. 2012).

Each service that an application requires needs a service broker. This broker is provided by the platform out-of-the-box for the common services (e.g., MySQL, MongoDB, MemCached, Redis). For other services, custom service brokers are created—

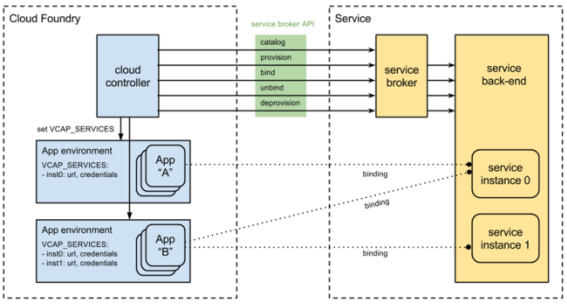

Figure 3: Cloud Foundry Service Brokering.

this includes IHE profiles web services similar to those described in (Ribeiro et al. 2011).

The platform is especially suitable managing micro services, which allows better componentization, development and testing processes, decentralized governance, resilience and maintainability. These services, especially when they are based on a RESTful architecture, are extremely easy to build, integrate, test, extend, and maintain, and are extremely adapted for mobile applications integration (Andry et al. 2011).

A service broker offers an API to fetch the catalogue of services (HTTP/S endpoints), provision service instances, bind or unbind services, and remove instances of these services.

2.4 IaaS Layer Abstraction

One of our requirements for the underlying PaaS layer was the ability to abstract the IaaS infrastructures models and vendors that the platform can use: private, public, on premise, or any hybrid combination, for the various healthcare solutions that need hosting. This approach has also been proposed by (Kolodner et al. 2011).

This is essential in countries where certain large public cloud vendors are not yet present. This is also crucial for on premise, limited footprint, and deployment of the platform for healthcare applications used in small and remote clinics, including in developing countries.

The advantage of abstracting the IaaS layer access through a common API is that there no need to have multiple versions of application code for each deployment model. The same code will work and be monitored the same way for all deployment models.

Another advantage is that the abstraction limits the situation of cloud IaaS vendor lock-in as described in (Sultan 2012).

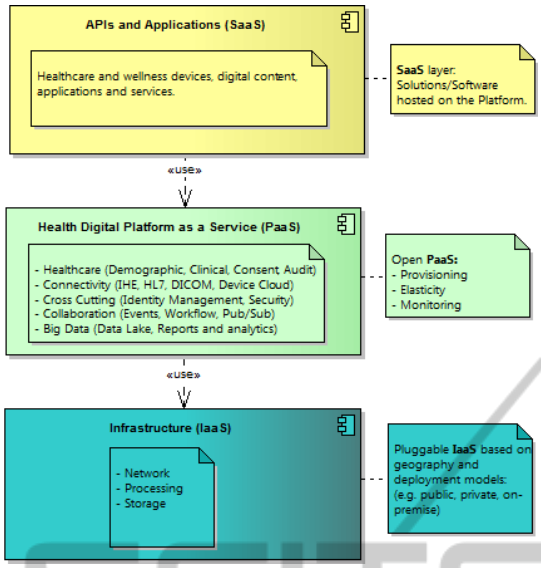

Figure 4: Open PaaS Hosting Healthcare Solutions.

2.5 Application Lifecycle Management

Good and efficient lifecycle management is important to produce and maintain high quality software. This is particularly important in healthcare where the patient life is at risk or a breach of privacy could occur as a result of poor quality software (Al-Khanjari 2014).

Figure 5: Application Lifecycle Management Steps.

The health platform being built leverages the cloud functionalities exposed by Cloud Foundry, then extends that functionality with custom stools, services and processes specific to healthcare. These extensions are used throughout the development lifecycle of applications.

Based on the type of application (healthcare or wellness), a development team might first use a set of interactive development environment (IDE) plugins and toolkits to either design the front-end of the application and/or look at a catalogue of web services to consume. Then the application can be developed using pre-determined best practices cloud patterns such as 12factors application patterns described in §2.2.

When the application is ready to be built and deployed, cloud resources from the IaaS layer pool are configured via the platform and the application is deployed. By default, especially if the initial expected volume is low, the number of application instances could be as low as one.

Developers and operators bind services to the application as required and the application monitored. The resource pool (CPU, memory, bandwidth, IP addresses, disk) can be adjusted as needed.

Patches and new versions of applications are deployed quickly without downtime in a continuous delivery model, mode where the platform router can toggle between an active version of the software, waiting for all new equivalent slices (web server, application server, back-end) and the new version waiting to be ready. Then all incoming requests are redirected to the new version components. The old components become idle and are recycled. This deployment model experiences no downtime and is completely transparent for the end-user since application instances can be fired-up and down extremely quickly. This model also offers a rapid rollback option if necessary.

The upgrade of the platform code itself can also be done without downtime.

Finally, deprecating applications can be completed and executed with total control, as monitoring tools offer a holistic view with fine-grain analysis of application usage, which is helpful with complex lifecycle processes in the heavily-regulated healthcare domain.

2.6 Application Stack Configuration

In our platform, the application stack is defined as a buildpack which provides framework and runtime support for our applications including all dependencies needed. For example, a Java buildpack might include dependencies for JRE, Spring framework, JDBC connectors, Servlet container,

logging and utilities.

Standard buildpacks are offered for various programming languages (e.g., Go, Java, Node.js, PHP, Python, Ruby, .NET) and they can also be customized. This is highly important in healthcare because large software assets have sometimes been developed over decades in various languages and are very difficult to re-write quickly due to their size and complexity.

A buildpack can either be described in a manifest file or specified during deployment as a github resource:

```
$ cf push my-new-phr-portal -b
git://github.com/acme-hc-dev/a-
buildpack.git
```

2.7 Monitoring

Operators can monitor instances of applications on the SaaS layer, health of the PaaS components, as well as IaaS resources, including the status of particular virtual machines (VM) where services and cloud foundry components are running. Examples of VM data points can consist of jobs, IP addresses, CPU load, memory consumption, swap statistics, and disk usage.

3 HEALTHCARE PLATFORM

On top of the generic open PaaS infrastructure, we are adding generic and cross-cutting capabilities not part of the original platform including:

- Identity management to allow customers, patients and consumers to be accurately and uniquely recognized by using an enterprise master patient index (eMPI) for patients and a lightweight directory access protocol (LDAP) based directory for healthcare providers and consumers.
- Security: authentication, authorization, and single sign-on, all critical to secure provider, patient, and consumer applications (Löhr H., Sadeghi A., Winandy M., 2010) and in certain cases, can be addressed by declarative proxification of these services (Faravelon 2013).
- Cloud-based, connected device management: device registration, discovery, routing, diagnostics, remote control, firmware provisioning, data collection, device-app-user pairing (we are currently supporting 6 million active consumer devices).
- Open cloud based clinical workflow collaboration capabilities
- Secure cloud-based big data store and analytics capability (e.g., to store patient's observations and genomic data)

We are also creating and exposing healthcare and wellness related services that applications can consume:

- IHE-based demographic, clinical, providers web services (e.g., PIX/PDQ, XDS, HPD)
- A virtual longitudinal healthcare record and associated services
- ATNA-based auditing services
- Consent and delegation services

Our healthcare platform offers high availability, scalability, privacy and security compliance with regulations (e.g., HIPAA, HITECH) and standards (e.g., NIST SP800-53, ISO 27001) using multi-tenancy, redundancy, 24/7 monitoring and operations, and disaster recovery.

3.1 User Management and Federated Identity

The applications can be used by different categories of users according to the context for which they are employed.

In healthcare applications, the users are patients or providers. The registration of those users is typically accomplished by healthcare professionals with strong permission management to secure authorized access for credited professionals and access to regulated and consented patient data records: aggregate virtual health record (VHR) coming out of electronic medical record (EMR) systems within an organization.

The patient does not have direct access to his/her clinical data part of the VHR, but can request that the data being exported to a personal health record (PHR) that the patient can then fully managed.

Within the same healthcare organization, patient identity will be reconciled via Patient Identifier Cross-Reference HL7 V3 (PIXV3) and Patient Demographic Query HL7 V3 (PDQV3) IHE web services fronting an enterprise master patient index (EMPI).

The EMPI stores a set of mapped fragment identifiers (medical records as well as internal Philips identifiers) for each patient.

For patient federation reconciliation across organizations, we are planning to use a mechanism comparable to a Cross-Community Patient Discovery (XCPD) web service.

In wellness and consumer lifestyle applications, the user is a consumer. The registration is usually

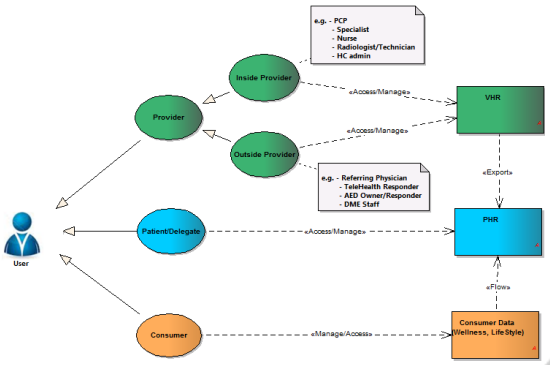

Figure 6: User Identity Federation and Data Access.

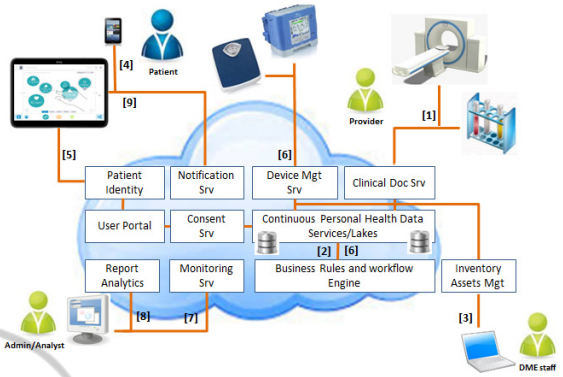

Figure 7: DME Home Care Delivery Use Case.

done by the consumers themselves using their preferred social account (e.g., Facebook, Google+, LinkedIn, Twitter) or via a traditional web form based registration where login and password credentials are associated to a basic profile information (e.g., name, email).

Subsequent accesses are done via the same credentials. Wellness and lifestyle data can be merged with PHR-based clinical data. The advantage of using social sign-on is that it is possible to collect automatically demographic and psychographic information such as email, address, name, geo-location, birth date, gender, interests, hobbies, friend lists, etc. This information, usually always up-to-date, can be reused over and over for various types of applications.

3.2 Platform Integration Use Case

In this example, a patient has been hospitalized and their demographic information and clinical data (e.g. observation, labs, scans, medication …) has been stored via the DHP clinical document service (1) on premise and optionally on the cloud (2).

After leaving the hospital, the patient needs some medical equipment (e.g., respirator) that is provided by a durable medical equipment (DME) company (3).

The patient receives a notification that an account on the DHP user portal is available (4).

The patient completes the registration and has access to their PHR extracted from their hospital clinical record (5).

Data coming from devices that the patient is using is collected, processed and routing via a rule based workflow engine to storage (6).

In parallel, admin staff can monitor the platform and associated applications (7), while analysts can use reports to improve the business process overall and help create new services and solutions (8).

Actionable events can trigger notifications to the patient or his/her care giver and specific reports are added to the patient's PHR and portal application (9).

4 DEVELOPMENT

Cloud based patterns and best service-oriented architecture (SOA) practices are used to externalize and store metadata (e.g., configuration data) for the applications, allowing them to be cloud-ready. Good practices include the 12factor app methodology:

- Using declarative formats for setup automation, to minimize time and cost for new developers joining the project
- Having a clean contract with the underlying operating system, offering maximum portability between execution environments
- Providing applications suitable for deployment on modern cloud platforms, eliminating servers and systems administration
- Minimizing divergence between development and production, enabling continuous deployment for maximum agility
- Designing applications that scale without significant changes to tooling, architecture, or development practices

The developers of these applications integrate and consume shared health-related and cross-cutting services such as identity federation, security, consent and access control, logging, auditing, all kinds of services for device-agnostic connectivity, as well as various others from a marketplace of services.

When applications deal with patients' private and consented data, complex security solutions have to

be put in place such as the ones described in (Dölitzscher et al., 2010), (Narayanan and Günes 2011), (Ermakova et al. 2013) and (Juels and Oprea 2013).

Most complex build, assembly and deployments steps to testing and staging environments are automatically handled by the platform and can be initiated by a developer instead of a dedicated build manager saving time and cost:

- Uploading and storing application definition files
- Examining and storing application metadata
- Examining and storing application metadata
- Creating a virtualized unit of execution for the application
- Selecting an appropriate execution agent to run the unit of execution
- Starting, monitoring and automatically restarting the application when necessary

The platform is being developed in small units of agile scrum teams (ten engineers and testers, maximum) and delivered in small, but regular increments, in sprints of three weeks with close to twenty teams and growing, spread around the globe in five different time zones.

5 OPERATIONS AND SUPPORT

With this new platform, the IT operations and support team have many additional tools to deploy the applications in production, monitor and maintain them.

The platform provides real-time analytics and alerts to monitor the health and status of the deployed applications (e.g., CPU, memory, disk, network, middleware components, completed and outstanding requests). These tools aggregate and log events as they are produced (e.g., from execution agents, VMs, routers and runtime resources).

The platform capacity can be scaled vertically by adding CPU, memory and disk, or horizontally by adding more VM instances for particular applications.

There are several ways to scale the platform for high availability:

- For components that support multiple instances, increase the number of instances to achieve redundancy
- For components that do not support multiple instances, choose a strategy for dealing with events that degrade availability

Maintaining an application deployed on a PaaS involves deploying patches, new versions, new buildpacks for OS and middleware upgrades, possibly new platform versions, and ultimately retiring applications.

The advantage of our foundation platform is that common middleware components and services (e.g., security, caching, data services and health enterprise services) are ready-made with no need to re-assemble the stacks and tiered components since a new VM is instantiated for new upgrade and patch.

For data services, the platform will configure and provide robust backup and restoration mechanisms, to provide robust availability and integrity of all data.

The healthcare platform operations cover many activities including but not limited to monitoring, release management, and incident response. The activities are defined by written and monitored operational-level and service-level agreements (OLAs/SLAs) and metrics. This is important since enterprise SLAs are simultaneously of high business value and technically challenging to implement (Lango 2014).

6 CONCLUSIONS

Adding new services and components on top of our digital healthcare platform enables new types of healthcare and wellness applications to be hosted, new types of data to be stored and new services to be exposed, increasing the complexity of the resulting platform.

However, it is not necessary to deploy all the components of the digital health platform in every configuration. For example, hosting a complex PACS solution within a hospital has few components in common with a set of services hosted in a public cloud that manages wellness data.

Nevertheless, the underlying PaaS layer needs a certain minimum of resources (e.g., memory, CPU, IPs, VMs) to operate. This could be an obstacle to a small footprint deployment if the required set of resources is too large.

In addition, the foundation platform that we are using needs to be large enough to support buildpacks for all existing assets required (e.g., programming languages, operating systems, framework), and will benefit greatly by migrating to the cloud.

We anticipate a re-architecting of the way containers are managed and applications are deployed (e.g., docker will replace warden soon) in the foundation platform which will be more efficient and portable.

ACKNOWLEDGEMENTS

We would like to thank Jeroen Tas and Dale Wiggins for their leadership and support in this long term and important project. Thank you also to Goutham Naval and Kyle Nguyen for helping bootstrap this new technology and Umang Nayyar for his great collaboration on this project.

Our appreciation to Ernest Angles Isern for sharing his knowledge in the domain of the device cloud.

We would also like to thank all members of HISS and DHP teams, in particular Eldo Issac, Ben Hallam, Brian Key, Vijayananda J. and Chad Evans for their contributions on this project. Thank you also to all business groups at Philips for their contribution and input to this new healthcare platform.

We are also very grateful to the Pivotal team, especially Yogesh Gupta, James Watters and Zach Brown for sharing their enthusiasm and experience of the Cloud Foundry platform.

REFERENCES

Al-Khanjari, Z., Al-Ani, A., Al-He for Establishing Privacy Domains in Systems of E-Health Cloud, in *International Journal of Engineering Research and Applications*, Vol. 4, Issue 7, pp.66-72.

Andry, F., Wan, L., Nicholson, D., 2011, REST-Style Architecture and the development of Mobile Health Care Solutions, in *Biomedical Engineering Systems and Technologies, series Communications in Computer and Information Science* (CCIS), pp. 301-311, Springer Verlag.

Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Ariel, Rabkin A., Stoica, I., Zaharia, M., April 2010, A View of Cloud Computing, in *Communications of the ACM*, Vol 53 Issue 4, pp. 50-58.

Bastião Silva, L., Carlos, C., Oliveira., J., 2012, DICOM relay over the cloud, in *International Journal of Computer Assisted Radiology and Surgery*, pp. 323-333., Springer.

Cloud Foundry, Cloud Foundry Components - 2014 - http://docs.cloudfoundry.org/concepts/architecture.

Dölitzscher, F., Reich, C., Sulistio, A., 2010, Designing Cloud Services Adhering to Government Privacy Laws, in *10th IEEE International Conference on Computer and Information Technology*, CIT, Bradford, West Yorkshire, UK.

Ermakova, T., Fabian, B., Zarnekow, R., 2013, Security and Privacy System Requirements for Adopting Cloud Computing in Healthcare Data Sharing Scenarios,in *Proceedings of the Nineteenth Americas Conference on Information Systems*, Chicago, Illinois.

Faravelon A. et al., 2012., Configuring private data management as access restrictions: from design to enforcement, In *Service-Oriented Computing*, ICSOC 2012, pp. 344-359.

He, K., Fisher, A., Wang, L., Gember, A., Akella, A., Ristenpart, T., 2013, Next Stop, the Cloud: Understanding Modern Web Service Deployment in EC2 and Azure, in *Internet measurement conference, ACM*, NY, pp. 177-190.

Juels A., Oprea A., 2013, New Approaches to Security and Availability for Cloud Data in *Communications of the ACM*, Vol. 56 No. 2, Pages 64-73.

Lango, J., 2014, Toward Software-Defined SLAs in *Communications of the ACM*, Vol. 57 No. 1, Pages 54-60.

Kolodner, E. et al., 2011, A Cloud Environment for Data-intensive Storage Services, in *3rd International Conference on Cloud Computing Technology and Science*, CloudCom, IEEE, Athens, Greece.

Löhr, H., Sadeghi, A., Winandy, M., 2010, Securing the E-Health Cloud, in *International Health Informatics Symposium*, IHI, Arlington, VA, USA.

Narayanan, H., Günes, M., 2011, Ensuring access control in cloud provisioned healthcare systems in *Consumer Communications and Networking Conference* (CCNC), IEEE.

Owens, D., 2010, Securing Elasticity in the Cloud, in *Communications of the ACM,* Vol 53, No 6, pp. 46-51.

The Twelve-Factor App, 2012 - http://12factor.net/

Ribeiro, L., Costa, C., Blanquer, I., Oliveira, J., 2011, On Demand IHE XDS Document Registries on the Cloud in *Conference: 29th International EuroPACS Meeting*, Vol 6.

Sultan, N., 2012, Making use of cloud computing for healthcare provision: Opportunities and challenges, in *International Journal of Information Management* 34(2): pp. 177–184.

Vaquero, L., Rodero-Merino, L., Buyya, B., 2011, Dynamically Scaling Applications in the Cloud, in *ACM SIGCOMM Computer Communication Review*, Vol 41, Number 1.