Journal of Cloud Computing
a SpringerOpen Journal

RESEARCH Open Access

Experiences in building a mOSAIC of clouds

Dana Petcu[1*], Beniamino Di Martino[2], Salvatore Venticinque[2], Massimiliano Rak[2], Tamás Máhr[3], Gorka Esnal Lopez[4], Fabrice Brito[5], Roberto Cossu[6], Miha Stopar[7], Svatopluk Šperka[8] and Vlado Stankovski[9]

Abstract

The diversity of Cloud computing services is challenging the application developers as various and non-standard interfaces are provided for these services. Few middleware solutions were developed until now to support the design, deployment and execution of service-independent applications as well as the management of resources from multiple Clouds. This paper focuses on one of these advanced middleware solutions, called mOSAIC. Written after the completion of its development, this paper presents an integrated overview of the mOSAIC approach and the use of its various software prototypes in a Cloud application development process. We are starting from the design concepts and arrive to various applications, as well as to the position versus similar initiatives.

Introduction

The story of mOSAIC (Open-source API and Platform for Multiple Clouds) starts in Spring 2009 when its idea emerged. The main challenges for Cloud Computing identified to that moment, as shown in [1,2], were application and data interoperability and portability, governance and management, metering and monitoring, as well as security. In the meanwhile some partial solutions addressing these challenges have proposed, however, according to a recent report [3], these challenges still exist. The key goal of mOSAIC is to offer a solution for application portability and interoperability across multiple Clouds. However, the complete mOSAIC' solution addresses partially also the other challenges, management, governance, and security, as will be revealed in what follows.

The mOSAIC' solution is a result of a multi-national team effort as part of a grant agreement with the European Commission in the frame of FP7-ICT programme [4] (details on the project web site [5]). The implementation has started in September 2010 and the final software was released in March 2013. The promises made in the early stage of development were described in the position paper 'Building a mOSAIC of Clouds' [6]:

1. Design a language- and vendor-agnostic application programming interface for using multi-Cloud resources and Cloud usage patterns.
2. Design a generic agent skeleton for representing various stakeholders, e.g. Cloud vendors and their resources, Cloud users of various types, and a collection of modules that can be used to adapt agent skeleton to support needed functionalities.
3. Design user-centric service level agreements, a Cloud ontology, and mechanisms for dynamic negotiation of resources based on multi-agent technologies and semantic data processing.
4. Build an open-source and portable platform for using Cloud services based on the proposed API and Cloud usage patterns.
5. Build proof-of-concept applications with a special emphasis on data intensive applications.

These scientific and technical goals were related with the time's lack of (a) common programming model for Cloud-oriented applications, (b) of tools for easy deployment of scalable applications and (multi)-Cloud-based services, (c) of standard representation for Cloud resources, (d) of adequate service level agreements and their dynamic negotiation, (e) of application portability due to different APIs for Cloud services from different providers.

A variety of reports on mOSAIC's particular software solutions have been presented at recent scientific conferences and in journal papers. The current paper intends to provide an overview of the full and integrated solution with exhaustive references to literature where details can be found. Therefore the main contribution of this paper consists in the high-level description of the mOSAIC's approach and the answers to the current challenges in

*Correspondence: petcu@info.uvt.ro
[1]Institute e-Austria Timişoara and West University of Timişoara, Timişoara, Romania
Full list of author information is available at the end of the article

the Cloud computing domain by the mOSAIC's technical solutions.

The paper is organized as follows. The first part is a description of the overall mOSAIC's approach for solving the current problems of using multiple Clouds. The second part concerns the mOSAIC's positioning in the landscape of Cloud computing services. The third part is dedicated to a discussion of the possible future developments and improvements.

The mOSAIC's approach

This section explains in details how the mOSAIC solution matches the key scientific and technical goals that were outlined in the introduction.

APIs and patterns

Several open API are already available (like jclouds, libcloud, OpenStack, most of them develop in parallel with mOSAIC) offering a management layer for the resources of same type from multiple Clouds (based on a common denominator of their APIs). However the services are restricted to a specific language (like Java), a specific architectural style (like REST oriented) or specific type of resources (like virtual machines).

Component-based programming

The component model provides a natural abstraction for programming and execution of Cloud applications, since is lightweight and flexible in terms of APIs, according [7]. However, component frameworks that are expected to provide design-time and run-time infrastructures in Clouds are few. A short preliminary analysis of the existing solutions was done for mOSAIC positioning purpose and is reported in [8]. This analysis revealed that a proof-of-concept implementation of a component framework for Clouds only for Java [9] was developed in parallel with mOSAIC.

The mOSAIC's API offers a simple way to develop components which run on the top of its platform. The programming model of mOSAIC is based on using loosely coupled components. A mOSAIC component represents an entity controlled by the user: the entity is configurable, exhibits a well defined behavior, implements application dependent functionalities and exposes them to other components. When an instance of a mOSAIC component runs in a Cloud environment, it consumes hardware or software resources, e.g. state-full resources hosted by Cloud service provider and accessible through dedicated APIs. The communication among moSAIC components takes place through message queues, e.g. using the AMQP [10] protocol or the Amazon's SQS [11].

The mOSAIC's basic component is the *Cloudlet* (first introduced in [12]). A Cloudlet is an event-driven and stateless component whose functionalities do not depend on the number of its instances at run-time (has a degree of autonomy). The Cloudlets can get automatic support for their life-cycle from the mOSAIC's platform including initialization, configuration and bindings to the needed hardware and software resources. Moreover, Cloudlets should be able to run in a Cloud environment independently from other components. Furthermore, Cloudlets are started, stopped, replaced, or multiplied at run-time for the application performance improvement. Multiple instances of the same Cloudlet are therefore expected to be supported by an application. Consequently, the elasticity concept, specific for Cloud computing, is applied at the level of application in the mOSAIC approach.

The Cloudlets are able to access Cloud services through Connectors. The concept of Connector is introduced to ensure the independence from the Cloud service interfaces. A Connector is a concrete class that abstract the access to Cloud resources and defines the set of events to which the Cloudlet should react; its behavior is similar to a remote procedure call, and it offers the functionality of the common denominator of a certain type of Cloud service. For example, in the mOSAIC library for Java there is only one Connector for key-value stores.

The Connectors access Cloud services using Drivers. The Drivers are implementing the Cloud services interfaces. They can be interpreted as wrappers of native resource APIs or uniform APIs, like OpenStack. These wrappers are able to send and receive messages from the mOSAIC's message queues.

The components of a mOSAIC application can be written in several different languages (Java, Python, Erlang, Node.js) and are able to communicate with each other using a component bus (similar to CORBA's one) and asynchronous communications (as being loosely coupled). In particular, Connectors and Drivers can be written in different languages. However, the Cloudlets that are expressing the behavior of the applications are calling the Connectors – therefore the Connectors are expected to be written in the same language as the Cloudlets (further details are provided in [13]).

A simple example

We assume that a software developer intends to built an application which is able to receive requests from the Web (e.g. an XML file) to perform an analysis of the document (e.g. XML parsing) and to store the results in a Cloud storage.

Such an application will be built easily using a predefined mOSAIC component which manages the HTTP protocol and offers the REST interfaces (the HTTP gateway, *HTTPgw*, in mOSAIC terms), and a Cloudlet that receives the XML files from the gateway, process them, and store the results in the Cloud storage using a key-value store Connector.

Once the application is developed it is possible to deploy it it and use a Cloud storage service offered by a Cloud Provider (e.g. Amazon S3) or a platform's internal component (like Riak-based service). The decision which one should be use can be taken at application deploy time (*not during the development*) and can be even dynamically changed at runtime. Moreover if the application needs to scale-up due to the high number of requests, the developer can just add at runtime new Cloudlet or other component instances to manage the newly incoming requests.

Event-driven programming

mOSAIC's API was designed to be event-driven. There are few implementations of event-driven approaches in Cloud computing, but the most known are Amazon's SNS, Microsoft's Azure, and the open-source Node.js.

The main reasons for such an approach are [14]: avoidance of expensive pooling on Cloud resources; opportunity to deal with an unlimited number of messages; adaptability that is naturally event-driven; rare changes in the state of long-running Cloud applications; integration with Internet of Things.

The event-driven approach has also drawbacks. The application developer needs to write the callbacks and the data cannot be provided with these callbacks due to the access rights. The states of a resource should be well defined to trigger a call to the API by the resource provider. Therefore, a dependence on the provider can be created for the callback. To overcome these problems, mOSAIC has proposed an abstraction layer (Cloudlets and Connectors) that allows the application developer to follow the concepts of event-driven architecture, while the low level components of the platform (Drivers) are treating the cases of demand-driven approach in the connection with the specific Cloud services. An interoperability component of the platform (between the Connectors and Drivers) acts as a proxy between the instances following the two different models of interactions (further details can be found in [15]).

Patterns

Currently, four pattern categories are used in mOSAIC: (1) programming patterns; (2) platform patterns; (3) service usage patterns; (4) application patterns.

The programming patterns are related to 'component-based' and 'event-driven' approaches. The patterns that are supported by the mOSAIC's platform are 'just-in-time-scalability' and 'event-based execution'.

The classical Cloud service usage patterns, as introduced in [16], are: end user to Cloud, enterprise to Cloud, Private Cloud, changing of Cloud vendors, Hybrid Cloud and so on. mOSAIC is mainly supporting the 'changing of Cloud vendors', and partially the 'Hybrid Cloud'.

Basic application patterns that can be used for quick application prototyping in mOSAIC were presented in an early paper [17]. Web, databases or application servers are supported. The proof-of-concept applications developed in the frame of mOSAIC project showed the approach usefulness for scientific applications. Therefore, Cloud related patterns for scientific applications were analyzed in details and are reported in [18].

Cloud agency

Role

The mOSAIC's Cloud Agency is a service for the deployment and execution of mOSAIC application. It is in charge for provisioning, from different providers, a collection of Cloud resources, which fulfill at best the user's requirements, to be consumed by mOSAIC applications.

The selection of the Cloud resources to be consumed is nowadays a challenging task for the developer or user of Cloud applications due to different business models associated with resource consumptions as well as due to the variety of features that the Cloud providers are offering. The IaaS commercial provider is interested in proposing a technological solution that is differentiating it from the others providers. This differences have the drawback of locking the customers as no alternative are provided. Also open source technologies for setting up Private Clouds are not compliant with each other. In this context, the Cloud Agency addresses the interoperability problem by proposing an uniform interface for accessing multiple IaaSs.

In the Cloud computing service market there are thousands of options which are different in terms of characteristics of the service, general terms and conditions and service levels that providers ensure. Their current Service Level Agreements (SLAs) use proprietary metrics that make difficult to evaluate properly each offer and to compare different offers among them. Moreover, the customer must trust twice its provider: because the agreed SLA, and because the provider's proprietary monitoring service. That is why the Cloud Agency aims at advancing the state of art of using the Clouds by providing a decision making support to the user for discovery and decision about the best Cloud solution that satisfies his requirements.

The Cloud customers need to detect under-utilization and overload conditions, and also to take decisions about load balancing and resource reconfiguration. In both the cases it is necessary to dimension the Cloud resource to avoid useless expenses and to not fail to satisfy the service requirements when workloads change dynamically. The Cloud Agency aims at providing a monitoring service that run on IaaS under the control of the customer.

Autonomic optimization of Cloud is widely investigated at provider side, but is not perceived by the customer as its own benefit, because it aims at maximize the provider's utility in terms of utilization of physical resource business improvement. That is why the Cloud Agency approach is based on autonomic agents, which enforce well defined policies to provide the perceived utility to their owners. The user is able to delegate to the Cloud Agency the necessary checks of SLA fulfillment, the monitoring of resource utilization, and, eventually, necessary re-negotiations.

The preliminary concept of the Cloud Agency architecture is detailed in [19], while the implementation layers to support Cloud applications were presented in [20]. The Cloud Agency is a step forward towards the implementation of the recent vision of Autonomic Cloud (as discussed in [21]).

Interfacing

Implemented as a multi-agent system, the Cloud Agency is based on asynchronous messaging as other mOSAIC software prototypes. The message-passing architecture was exposed in [22].

The Cloud Agency can act as a standalone and independent component or as an integrated platform component. In the first case it can be used to book Cloud resources and eventually monitor and reconfigure them (scale up, scale out, change providers), without the need of other mOSAIC components. Several user-friendly and programmatic interfaces can be used for interaction (a list and their descriptions can be found in [23]). In the second case, the Cloud Agency offers services to the core components of the platform (e.g. in the case of reconfiguration at application level). A RESTful interface can be used for the interaction [24]. It is compliant with OCCI.

While multi-agent based Cloud management architectures or frameworks were proposed before mOSAIC (like in the proposal exposed in [25]), we consider that mOSAIC's Cloud Agency is the most complete implementation of the concepts that has been tested in the context of complex scientific and commercial applications.

Resource management: provisioning and monitoring

The Cloud Agency provisions resources which should be consumed by the applications. Through its interfaces, detailed in [26], the Agency Client can start a Call-for-Proposal, based on a component description of the application and the policy specifications (a HTTP POST message embedding an SLA is sent to the Agency). The availability of a result triggers an event (using a HTTP POST message), after which the Cloud Agency Client is able to accept or reject the proposal.

The brokering of the best collection of Cloud resources has been modeled as a multi-criteria optimization problem, with hard and soft constraints that can be included by the user in the Call-for-Proposals, as it is described in [27]. With regards to a computing service, such constraints can be required for service properties, like CPU architecture, minimum amount of memory, CPU speed, I/O speed or number of cores. The level of the service availability can be set over a threshold. Multiple objectives can be defined by the user to choose the cheaper proposal and/or to optimize the performance of I/O-bound or memory-bound application.

Beyond the provisioning role, the Cloud Agency has also other resource management functionalities, like monitoring, that is related to the parameters specified in SLAs (monitorinf the quality of service). Details about this role implementation can be found in [28,29].

Vendor modules

A Cloud provider offer is represented in a brokering or negotiation process by a Vendor Module; details about the module architecture and interaction with the Cloud Agency were exposed in [30]. A simple template was designed to offer a mechanism for new Cloud providers to connected their services to the Cloud Agency; the template was first documented in [31].

Until now mOSAIC' software repository includes Vendor Modules for more than ten Public Clouds. Among these mOSAIC supports well known providers like Amazon, Rackspace [32], and GoGrid [33], as well as European Cloud providers including Flexiant [34] (UK), CloudSigma [35] (Switzerland), NIIFI [36] (Hungary), Arctur [37] and Hostko [38] (Slovenia), latest two using VMware's vCloud [39], respectively OnApp [40]. Moreover, Private Clouds built by using open-source technologies, like Eucalyptus [41], OpenNebula [42], CloudStack [43], or OpenStack [44], can be also represented in the brokering process by their corresponding Vendor Modules and managed by the Cloud Agency's uniform interface.

User-centric SLA management and dynamic negotiation

SLA management

Service Level Agreements (SLAs) are the basics of a common language for agreements between the Cloud clients and the Cloud service providers. Due to the self-service approach, typical for Cloud computing, a SLA for a Cloud service has from user perspective a relevant role in defining what the service effectively grants. mOSAIC supports SLA both at brokering level and at API level. In the first case, SLAs are used for the brokering mechanisms through the Cloud Agency and through a SLA client with respect to provider's offer. In this second case, mOSAIC acts as an SLA provider: the offered services are enriched with SLAs.

The API offers to the application developer a framework which helps in building custom SLA, as well as in seamless integration of their management in service provisioning. At state of art few frameworks exists that offers such kind of functionalities: the one produced by SLA@SOI [45-47], which proposed a solution for building SLA managers to be integrated in service oriented architectures, or WSAG4j [48] which is a Java library compliant with the WS-Agreement standard (that defines protocol and format for SLA representation and management). Such frameworks are complex and expected to be integrated by Cloud providers: SLA are defined by the provider and the users can access only a set of predefined templates.

The SLA management in mOSAIC is considered to be different from such predecessors in its concept: the main goal of mOSAIC' SLA framework is to enable a developer to easily integrate a single application with an SLA life cycle, so instead of offering a single and static general purpose solution for SLA management for any application, a set of micro-functionalities is offered to be integrated with the application in order to build up a dedicated solution for the application developer problem. Due to the component-based approach of the API, it is possible to build up applications enriched with user-oriented SLA management, from the very early development stages. Example of such microfunctionalities are the *SLAgw* which offers a REST-based interface to submit and sign SLAs, the *SLAstore* which maintain the SLA life cycle or the *SLApolicy* which is adopted to automate the enforcement of SLA policies.

Following such approach SLAs can be defined both by the developers (offering SLA templates like with other solutions) or defined by the users (following the standard WS-Agreement format); in such latter case it is up to the application to parse and eventually accept or refuse the submitted SLA. mOSAIC' SLA framework offers the tools to access the submitted request and templates for building custom *decision components*, which have the role of making decisions on the basis of the SLA submitted. Such user-centric service level management is further discussed in details in [49].

Note that the SLA parameters to be supported in such solution are strictly dependent on the application and their management is delegated to the application developer. The SLA framework was applied further for the management of security-based SLA (in [50-53]) and has been integrated in simulation engines able to predict the evolution of the developed application [54,55].

SLA-based brokering and negotiations

The mOSAIC's brokering mechanism is an intermediary between the resource consumers and the resource providers. The best SLA from the point of view of consumer is identified [56]. The policies that can be used are presented in [57], the basic one being the 'lowest cost'. SLA-based brokering mechanisms for 'lowest cost' strategy were detailed in [58].

Exploiting the event-driven architecture, a SLA condition violation triggers an event that can lead to a reconfiguration. A reconfiguration mechanism was therefore conceived, and it is based on rules and a reasoner [59].

Assuming that the Cloud providers are willing to negotiate the costs of the resources, a more complex mechanism with stages for negotiations can be conceived, as the one presented in [60].

Semantic processing and ontology

Semantic engine

The Semantic Engine is a mOSAIC component helping the user in selecting APIs components and functionalities needed for building new Cloud applications as well as in identifying the proper Cloud resources to be consumed. It introduces a new level of abstraction over the Cloud APIs, by providing semantic based representation (in the OWL language of the Semantic Web) of functionalities and resources, related by properties and constraints. The detailed architecture of the solution is presented in [61].

Using the Semantic Engine the developer of Cloud applications can semantically describe and annotate the developed components, specify application domain related concepts and application patterns, potentially using application domain ontologies, as explained in details in [62].

The Semantic Engine overcomes the syntactical differences between Cloud services, resources or their programming models. Automatic analysis of Cloud Vendor APIs is therefore possible, as demonstrated in [63]. Moreover, the semantic representation of Cloud APIs combined with automated algorithmic concept recognition in object-oriented code, augmented with structural based matchmaking techniques, can be a strong basis for porting existing applications towards Clouds [64].

The semantic techniques are used for describing application requirements. The Semantic Engine infers the infrastructural requirements from the application description and from other information, and produces a vendor agnostic SLA template [65].

Cloud ontology

While several Cloud ontologies were developed before mOSAIC (like the one proposed in [66]), the mOSAIC's one is built upon existing standards and proposals analysis through annotation of documents, as described in depth in [67]. It is used in the mOSAIC's semantic processing.

The mOSAIC's Cloud ontology has been developed in OWL. It has been populated with instances of Cloud provider APIs. The knowledge base can be extended with new Cloud provider APIs in the future. The initially proposed ontology was first exposed in [68]. Later on it was augmented with services specific terms – a list of enhancements is presented in [69].

Integration platform

For the purpose of this paper, the collection of individual components that represent mOSAIC's proof-of-concept prototype solutions are depicted in Figure 1. Individual components, like API implementations, application developing tools, vendor modules and so on are part of the integration platform (named here the mOSAIC's PaaS).

Core components of the software platform

Core components are aiming to enable the run-time and deployment functionalities offered by the platform. These include: mOS, Deployer, Container, Components-of-the-Shelf (COTS) and Drivers. They are responsible for the platform control, scheduling, scaling, monitoring, application deploying and so on.

mOS is a customized Linux kernel running in virtual machines. It is used to host and control the platform and the application components. The Deployer is a core component which deploys software modules on mOS: the packaged component is retrieved by the Deployer from the named location and installed into an appropriate execution environment inside the instantiated Virtual Machines. A Container is a component which hosts Cloudlets, and is responsible for meeting the requirements of elasticity and fault tolerance. The Drivers access specific API of external Cloud services, and are built, for example, for message queue mechanisms like RabbitMQ [70], for key-value stores provided by Amazon S3 [71] or Riak [72], or for distributed file system HDFS [73].

The functionality of the core components was described for the first time in [74]. Aspects like reliability or fault-tolerance support are treated in later papers like [75].

Commercial off-the-shelf components

A number of software components were adapted, so that they can interoperate with the mOSAIC platform and facilitate seamless application development and deployment. Examples of components off-the-shelf (COTS) that

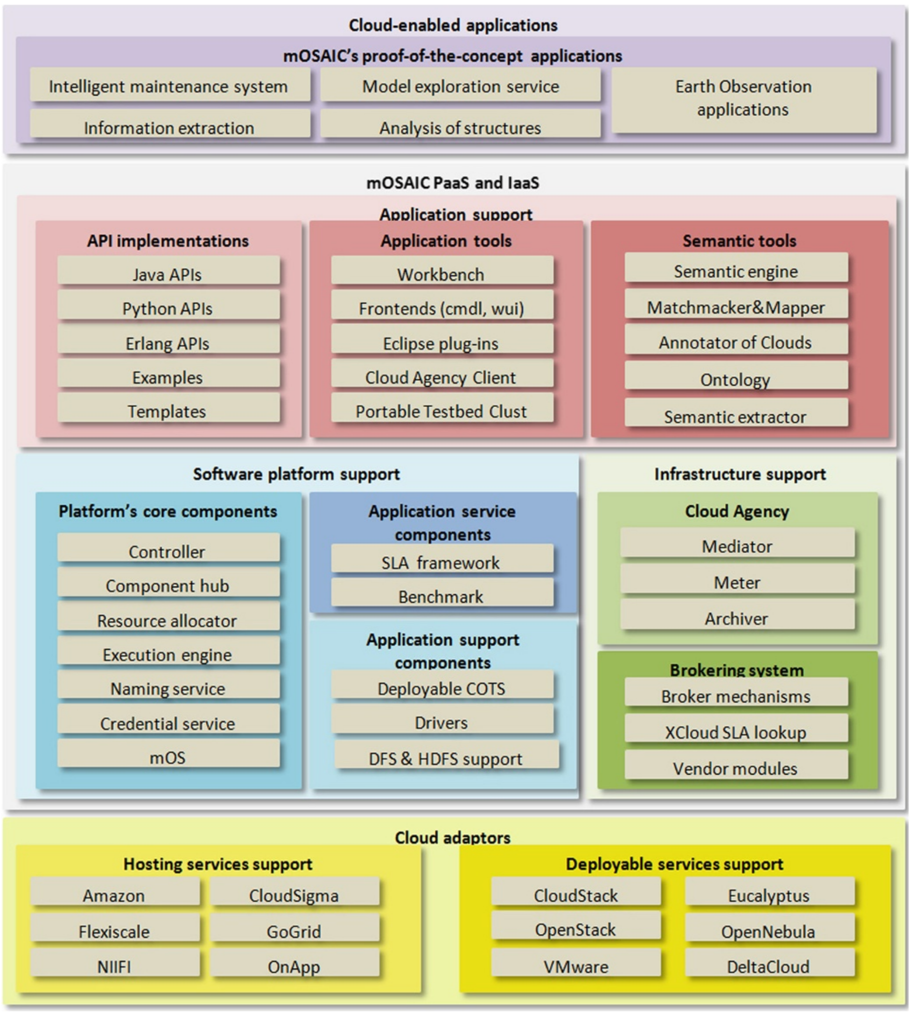

Figure 1 Component view of the integration platform.

are currently available are: RabbitMQ Server, Riak Server, CouchDB server [76], Jetty web application server [77], MySQL server.

In mOSAIC, COTS are viewed as Cloud software resources, based on open source technologies and reusable. These components are deployed as any other component and can be managed, monitored, and accessed via specific Drivers.

Application development and deployment tools

Several tools are offered to enable the seamless development of new applications or the deployment of the existing applications. Eclipse plug-ins, for example, allow the development, deployment, debugging and control of the components written in Java. A Web interface and a command line interface are available to monitor the status of the deployed components (including analysis of the logs) and to control the life-cycle of the running applications. Configuration tools and editors are provided so that, for example, it is possible to build Call-for-Proposals, which may be directly submitted to the Cloud Agency.

The Portable Testbed Cluster (PTC) is a specific component that allows for application testing on the developer's desktop computer. The PTC simulates a Cloud environment using VirtualBox [78]. Its also uses a simple resource allocator, a credential service (which stores the credentials for a specific user for a specific Cloud provider), and the storage of the application components in a Web accessible location. Through its graphical interface the PTC allows the seamless movement of the locally developed application into a Public or Private Cloud.

Open-source and portability of the integrated platform

A large part of the integrated platform is offered as open-source in the mOSAIC's Bitbucket collection [79]. Tables 1 and 2 map the components to the corresponding repositories. The open source part makes it possible to design and execute a variety of Cloud applications.

The platform is theoretically deployable on any Linux-based virtual machine. mOS needs to be first installed to make available the core components. For several providers (e.g. Amazon, Flexiant), virtual machine images, with mOS already installed on top, are publicly available.

The fact that the mOSAIC's platform is deployable makes the difference from other PaaS, like Google Application Engine [80] or Microsoft Azure [81]. An open-source and deployable PaaS, developed in the same time with mOSAIC, is the VMware's Cloud Foundry [82]. Compared with this one, an advantage is the complete openness of the mOSAIC's API.

Platform-as-a-Service characteristics

As discussed earlier, by using mOSAIC, the developer of a Cloud application can postpone the selection of a Cloud infrastructure provider, from the development to the deployment phase. Through the usage of the application tools, a seamless deployment of the application in various Public Clouds is possible after its development and debugging in a Private Cloud or on a local computer. This is more than a Platform-as-a-Service can currently offer – usually the application is deployed immediately on the infrastructure that the PaaS owner provides.

A classical PaaS is not exposing the infrastructure services that are used, while in mOSAIC its user is aware and able to control the resource services. From this point of view mOSAIC distinguishes from other PaaS, as being a portable and lightweight management software for IaaS.

Proof-of-concept applications

The concept of a mOSAIC Cloud application

A mOSAIC compliant application is built from loosely coupled components. Its execution is expected to have no limits in time. It may be expected that some of the components would be elastic, i.e. able to scale up and down in number. The most common cases of Cloud applications are Web applications, which fully fit into this behavioral model. However, the mOSAIC API and platform are suitable for building and deploying also other applications which can have the same behavior.

There are two basic scenarios of using the mOSAIC set of solutions:

1. Developing a new application from scratch. In this case the developer has the possibility to use most of the software tools provided by mOSAIC, starting with the API implementation and the application tools. The Semantic Engine can be used to find proper design patterns. Application tools can be used to prepare a Call-for-Proposal for the Cloud Agency and to approve one of its responses. Software platform services are responsible to deploy, control and monitor the executing application.
2. Migration of an existing application. In this case the application owner may be interested in finding a proper place to deploy the application, and the Cloud Agency is the main tool which is used in the process. The application owner may use the application tools to properly describe the application and to make a Call-for-Proposal to the Cloud Agency. Only a part of the platform is used to control the application after its deployment.

Benefits of using mOSAIC

The main reason of using the mOSAIC solution may be its vendor-agnosticity. The application developer and owner can select at run-time the Cloud services to be consumed (usually this decision need to be made at design phase).

Table 1 Open-source codes in Bitbucket repository (Part I)

Component	Description	Sub-repository
	API implementations	
Java API	Provides the developer with an asynchronous API for managing a customized Component and allowing the access to various data sources and backends	Mosaic-java-platform
Python API	Similar with Java API, but for Python	Mosaic-python-platform
Erlang	Tools for Erlang	Mosaic-erlang-tools
NodeJS	NodeJS implementation	nodejs-libraries
DFS Java Connector	Java Connector for distributed file systems	Mosaic-java-connectors-dfs
Realtime feeds	Allows interested users to receive live updates of various ATOM feeds, via a Web interface	Mosaic-examples-realtime-feeds
	Application tools	
Workbench	A set of functionalities implemented inside the EclipseRCP that permits to have a fully functional Eclipse workbench to work with the mOSAIC platform	Mosaic-workbench
Frontends	Allow the user to interact with the actual tool implementations (the backends) via various interfaces like CLI (Command Line Interface) or graphical UI's	Mosaic-node-wui
Eclipse plugins - Editors	Provides the end user a way to edit the configuration files in a completely visual way	Mosaic-workbench
CA Connector	Cloud Agency Connector	Mosaic_ca-connectors
Portable Testbed Cluster	Local virtual cluster environment that simulates a IaaS	ftp.info.uvt.ro/mosaic/ptc/
	Application service components	
SLAgw	SLA REST interface	Mosaic-java-SLAgw
SLA-components	SLA storage management component	components-SLA
Benchmark-SLA	SLA Policy Component	components-benchmarks
Benchmarks	Java benchmarks	Mosaic-java-benchmarks
	Vendor modules (in mosaic-vendormodule-*)	
Amazon	Amazon vendor module	-amazon
CloudSigma	CloudSigma vendor module	-cloudsigma
Eucalyptus	Eucalyptus vendor module	-eucalyptus
Flexiscale	Flexiscale vendor module	-flexiscale
GoGrid	GoGrid vendor module	-gogrid
NIIFI	Vendor module for Hungarian IaaS	-niifi
OpenNebula	OpenNebula vendor module	-opennebula
OpenStack	OpenStack vendor module	-openstack
PTC	Module for mOSAIC's PTC	-ptc
VMware	VMware 's vCloud vendor module	-vmware

A side effect is the possibility to migrate applications from one Cloud provider to another. Therefore mOSAIC can be used to port applications between Clouds. Other reasons for using mOSAIC may be more technical:

1. Ability to ensure the elasticity at component level (usually done to a lower level of granularity, at virtual machines level).
2. Integration in one set of solutions of application development tools with deployment and control tools, as well as with Cloud brokering mechanisms.
3. Open-source technology that allows extensions as needed for special applications or embedding of other technologies.
4. Deploy-ability that allows to use on-premises resources in development phase and to build Private or Hybrid Clouds enabled applications.

Demo applications for the API usage

Simple applications, like the 'Hello world', producer-consumer, ping-pong messaging and so on, are provided for the Java implementation of the API. The example

Table 2 Open-source codes in Bitbucket repository (Part II)

Component	Description	Sub-repository
	Platform's core components	
Controller	Allows the developer to observe and control the running components (either Cloudlets, Drivers, Resources, etc.)	Mosaic-node
Component hub	Intermediates communication between the components, and other various needed services (eg. logs)	Mosaic-node
Resource allocator	Resource provisioner based on existing credentials	Included in PTC
Execution engine	Container: Component responsible for instantiating linux containers (LXC) from mosaic bundles (containing all required data for running an application)	Mosaic-execution-engine
	Agent: Running inside the Containers providing application setup, startup and monitoring	Mosaic-execution-engine-agent
Naming service	Allows the registration of new components and their discovery by other components	Mosaic-node
Deployer	A Python RPC implementation based on JSON-RPC2, providing introspection, Unix socket based protocol, standard Http protocol support and a simple CLI application implementing the protocol	mjsrpc2
Packager	Packaging utilities and repositories	Mosaic-packages-repositories
Scheduler & Scaler	Scheduling and scaling components	Mosaic-scheduler
Credentials service	Provides secure access to various credentials or secret tokens needed by various libraries or components to access external resources	Mosaic-credentials-service
mOS	mOSAIC operating system: the system includes the platform's core services	Mosaic-mos
	Deployable COTS and Drivers	
RabbitMQ component	Customized variant of RabbitMQ which makes it behave like a managed component	Mosaic-components-rabbitmq
Riak component	Customized variant of Riak which makes it behave like a managed component	Mosaic-components-riak-kv
CouchDB component	Customized variant of CouchDB which makes it behave like a managed component	Mosaic-components-couchdb
HTTP Gateway component	Intermediates HTTP requests between clients on the Internet and components handling those requests. Provides routing and load balancing	Mosaic-components-httpg
mHTTP Gateway component	Routes HTTP messages on queue and enable access to a key value store. Renders HTML pages	Mosaic-components-mhttpgw
AMQP driver component	Message Queue driver for the AMQP protocol	Mosaic-java-platform
Riak driver component	Key value driver for the Riak component	Mosaic-java-platform
HDFS driver component	Distributed File System driver for Hadoop	Mosaic-java-drivers-hdfs
MySQL component	A component for handling (starting/stopping) a MySQL database engine	Mosaic-components-mysql

applications as well as various Cloudlet templates are available in the open-source repository.

A more complex application that combines COTS and developed Cloudlets, is a so-called real-time-feed application. It is small a RSS feed alerter. A classical implementation supposes a poll mechanism that constantly fetches content and makes a comparison with the previous version of the retrieved content to see what is new. On contrary, the demo application implements a push method which identifies the changes and announces them to the event listeners. It uses Twitter' streams of updates to subscribed applications. A first-time description is available in [83].

Proof-of-concept scientific and commercial applications

In the context of mOSAIC project, a variety of applications were developed. These include:

1. an Earth observation application where IaaS is procured for processing satellite data in emergency situations (e.g. earthquakes);

2. an Intelligence Maintenance System allowing maintenance of devices from different industrial scenarios through early diagnosis of faults in critical components and real-time monitoring of key variables;
3. a Model Exploration Service, an online service to run agent-based simulation, requiring scalability;
4. a port to the Cloud of a legacy application for the information extraction from scientific papers;
5. a port to the Cloud of an engineering application for analysis of structures under static loading, with the intention to make it available on the Internet without the need for special licenses or environmental settings.

The Earth observation application prototype dealing with big sets of large-sized data is based on GMTSAR, an open source processing system. It uses the mOSAIC Cloud Agency as independent component for the provisioning of computational resources to create Virtual Clusters to run an OpenGIS Web Processing Service server and a Hadoop framework. The main benefit of using the mOSAIC approach is related to vendor-agnosticity. The prototype is described in [84].

The Intelligent Management System intends to manage easily and rapidly large amounts and continuous streams of information. It has been written using the mOSAIC API specifications. Two main groups of components are used: components provided by the platform (message queues, storage systems, HTTP gateway, etc) and specific components implemented as Cloudlets (for sensor data management, knowledge extraction dispatcher, etc). The main benefits of using mOSAIC are: the elasticity of the application depending on the required computation capacity, and the usage of fault tolerance features. A preliminary description is provided in [85].

The Model Exploration Service is an online service to run agent-based simulations in the Cloud. Cloud resources are used to run large parameter sweeps of the models. The initial application has already run using Amazon's AWS. The porting of the application by applying the mOSAIC approach has not involved only the change of calls from the Amazon API to the mOSAIC API. The internal architecture of the application was changed to adopt mOSAIC design principles (the Cloudlet model) to facilitate scalability, portability and autonomous reconfiguration. By avoiding vendor lock-in, the new version of the application can be moved across 'Cloud borders', or the application can be Multi-Cloud in the sense that certain parts run on different provider infrastructures. The application architecture was first presented in [86].

Information extraction from scientific papers is a part of the ReReSearch project that aims to build a knowledge base about research. It is a computationally expensive task and, as the number of papers to be processed varies considerably in time, there is a need for elasticity of the Cloud (the preliminary study [87] lists the requirements). The legacy Python code implementing the extraction algorithms was split into components and wrapped into Cloudlets. These Cloudlets, managed by the mOSAIC platform, make the application elastic and portable. The Cloudified extractor is exposed as a RESTful web service which can be used as needed by ReReSearch control system. Implementation details are exposed in [88].

The Analysis of Structures under Static Loading application is based on a specific Finite Element formulation, which allows the modeling of a desired structure by beam elements. It is based on the NoDeK software which is written in Matlab and used by small number of construction engineers, due to the required Matlab licenses for running the simulation. The application has scalability problems and its user-friendly interface could be considerably improved. There are several reasons to use Cloud computing technology, e.g. offering the application as a service has the target of widening the customer base. The reasons of selecting mOSAIC is related to the expectations of faster calculation in Clouds than on the local desktops, as well as to deal with the dynamic change in the number of concurrent users. The design of the Cloudified version of the application is available in [89]. Matlab-based Cloudlets were designed and are presented in further details in [90]. The use of Semantic Engine usage in the development phase of the application is exposed in [91].

Benchmarking

A mOSAIC application is developed without taking into account the target provider, however when the application needs to run and consume resources, the choice of the infrastructure provider may significantly affect the application performances and costs of using the infrastructure. The evaluation and prediction of such performances for applications is a complex task, due for example, to the elasticity offered by Cloud resources, or the high number of layers involved.

In order to face the problem of choosing appropriate infrastructure provider, mOSAIC provides a Benchmarking Framework. The framework contains a set of components that can be used in order to setup a custom benchmark which measures the performances of the target application under well known workloads. Such benchmarks are built "ad-hoc" for each different application, even if a set of stable application for common resources and application are available and can be used as kernel benchmark to compare different providers [92]. Moreover, through the adoption of simulation techniques it is possible to use benchmark results to predict the behavior of application in different execution conditions, as proposed and demonstrated in [93].

Scientific applications support

While being partially an open-source solution, it is expected that mOSAIC is interesting especially for the academic communities. As pointed above, mOSAIC has prove its utility for several scientific applications with different requirements. Therefore, the opportunity for mOSAIC to be a science facilitator was investigated recently in [94]. An early study of migration from Grid to Cloud using mOSAIC concepts is provided in [95] for a platform intended for high education activities in Earth observation. The support for scientific legacy applications through self-configuration is exposed in [96]. For the particular case of engineering application, a comprehensive study can be found in [97].

Beyond the promises

Security

Security is a critical issue for the Cloud adoption. While mOSAIC has not promised initially to deal with this topic, in order to offer a complete solution, the problem of security in multiple Clouds was studied in order to define, develop and adopt a mOSAIC-specific security approach. Taking into account the target of mOSAIC for multiple Clouds, the available solutions for Cloud Federations were analyzed in deep in [98], establishing the basic requirements. The research activities acquired experiences from previous work related to Cloud and Grid integrations, as exposed in [99]. The main security problem that was treated is related to the access control solutions, as presented in [100]. The mOSAIC's SLA framework was integrated into an automated access control mechanism for the Cloud and Grid, as proved in [53]. The FP7 project SPECS (Secure Provisioning of Cloud Services based on SLA management [101]), starting in Autumn of 2013, will continue to enhance the mOSAIC's SLA framework.

Due to the Cloud elasticity and auto-scalability features, denial of services was considered one major threat. Therefore denial of service attacks, specially targeted to Cloud systems, were were studied in [50,52]. A mOSAIC based solution for protecting Cloud applications against such attacks was proposed in [51]. Finally, a special service was designed based on SLAs: Intrusion tolerance as a Service [102].

Another research direction considered the secure authentication negotiation in Cloud. The first results are exposed in [103].

Governance

Cloud governance comes as the next development step after Cloud management. It provides the ability to set policies within the environment in order to ensure the system's wide security, privacy and compliance. It is provide also the business level missing in Cloud management solutions. While the Cloud management is providing an execution environment, according [104], the Cloud governance is in charge of decision making in order to achieve objectives that meet its customers' needs.

As shown in [105] the multi-agent systems are fit to build a Cloud governance solution due to the autonomous nature, fault tolerant behavior and ability to self-organize. Therefore the mOSAIC which is based partially on multi-agent systems provides a convenient environment to study the merge of Cloud management and Cloud governance. To do so, the requirements in order to achieve Cloud governance in mOSAIC were established in [106], most importantly being the ones related to service lifecycle control and governance bus.

Cloud service lifecycle in a Multi-Cloud needs to take into account the distribution of services across different provider sites and the ability of services to scale. The core aspects of a service lifecycle, including service template, offering, contract, provisioning service, runtime maintenance, and end of service, are extended in the mOSAIC's governance architecture, first time presented in [107].

A Cloud governance bus needs to handle messages, security, exceptions, protocol conversion and to provide an adequate level of quality of services. The bus proposed in [108] implements enterprise integration patterns as well as data integration services which enables easy access to datastores.

Particular attention was given in mOSAIC to the data services as part of the proof-of-concept applications which are dealing with big data (static and streaming). Data-stores and the appropriate services were studied and developed in conjuction with the mOSAIC's Cloud governance solution [109,110]. Another issue that was treated in the same context is the security of the data [111].

The Cloud governance can lead to an unitary ecosystem, where applications can be easily created, managed, discovered and can easily interact one another. This is aligned with the idea of InterCloud and Cloud Blueprint, introduced in [112]. The grounds for such an ecosystem based on mOSAIC's approach were put in [113].

Resource management: scheduling and application monitoring

Scheduling mechanisms are widely used in distributed systems. However the particularities of Clouds impose a reconsideration of these mechanisms. In order to come with a practical offer, the scheduling problem in Clouds and Multi-Clouds was studied. An initial proposal for the particular case of workflows and based on multi-agent systems was presented in [114]. Starting from this proposal and exploiting the capacity of multi-agent systems for regeneration, a self-healing scheduling mechanism was later on presented in [115].

A large category of Cloud applications are long-running and their high availability is essential. In this context and taking into account the design of mOSAIC application based on components, in [116] is proposed a scheduling mechanism of replicated components aiming to reach the objective of highly availability despite multiple faults in Multi-Cloud. Another scheduling mechanism based on cost constraints was presented in [117], while a novel P2P scheduling scheme has been introduced in [118].

Monitoring services are usually referring to the infrastructure delivery indicators and are in relationship with service level agreements and quality of services. However, the user is highly interested in the behavior of particular applications deployed in Clouds. Therefore an important topic should be the Cloud application monitoring. Taking into account the component based architecture of the mOSAIC applications (including the communication system as a component), the component monitoring can be considered. This approach for the monitoring services was first presented in [119].

Model-driven engineering

In mOSAIC a Cloud application consists of loosely coupled components, which in particular can wrap legacy software. In this context, the paper [120] investigates the possibilities to introduce a model-driven architecture which support composition, customization, flexibility, maintenance and reusability of Cloud application components in the particular case of scientific and engineering applications.The approach is illustrated through the design and operation of the application for analysis of structures under static loading.

A methodology, named MetaMORP(h)OSY, that uses model-driven engineering and model transformation techniques to analyse Cloud services was introduced in [121]. Due to the complexity of the systems to analyse, when modeling profiles are built with MetaMORP(h)OSY, the mOSAIC Ontology is used as being able to specify Cloud domain-related properties. Following the methodology, a proof-of-concept for a particular Cloud use case is provided in [122].

Related projects, prototypes or applications

Related projects

mOSAIC has a strong relationship with two national projects: the Italian Cloud@Home [123] and the Romanian AMICAS [124].

The primary goal of Cloud@Home is to implement a volunteer Cloud, by which both the commercial/business and the volunteer/scientific viewpoints coexist. Several topics that are treated by Cloud@ Home and are not primary targets to mOSAIC have lead to common proposals. Two topics are in this context relevant: performance management exposed in [125], and quality of services exposed in [126]. Moreover the SLA-based mechanisms for brokerage are common to mOSAIC and Cloud@Home [127]. CHASE [128], Cloud@Home's Automatic Service Engine is designed to optimize the scheduling of virtual machines in a Cloud environment based on the a performance prediction service and a forecast service.

The primary goal of AMICAS is to offer a solution for Automatic Clouds. Opposite to mOSAIC which is targeting the Cloud users, AMICAS is targeting the Cloud providers, intending to offer them an easy manageable middleware (Cloudware) for Multi-Cloud. It starts from the mOSAIC's software platform and enhances it with facilities of interest to Cloud providers. An important topic that is tackled by AMICAS, using the mOSAIC experimental platform, is the programmability of services for multiple Clouds. The steps to reach a high level of programmability were discussed in [129], while the programmatic management of services from multiple Clouds using mOSAIC was described in [130]. Another subject is the auto-scaling mechanisms, essential in Cloud computing environment. Currently most of such mechanisms that are used by Cloud providers are centralized. Taking into account the perspective of the Multi-Cloud, such centralized approach is not appropriate. A decentralized auto-scaling mechanism was therefore proposed in [131] for the case of homogeneous systems, and was extended for the heterogeneous systems in [132]. Theoretical analysis of the background algorithm correctness is presented in [133].

mOSAIC's software platform is used as a Multi-Cloud resource management middleware in another current project funded by the European Commission: more precisely, it plays the role of run-time environment in the model-driven engineering project named MODAClouds [134]. Its role in the architecture is explained in the early position paper [135].

Moreover, the Earth Observation application can be viewed as a preliminary study for the Earth Observation Use Case scheduled in the frame of the Helix Nebula project (Science Cloud initiative funded by the European Commission [136]).

Related prototypes

As stated earlier, mOSAIC is interested to support scientific applications running in Clouds. A large category of scientific applications are based on parallel computing simulations.

mJADES is a prototype for concurrent simulations in Clouds, using the mOSAIC's SLA framework. It is the result of Italian PerfCloud project (building an environment for IaaS provision based on Cloud and Grid integration). Its architecture is explained in [55]. Using this prototype several performance prediction studies of Cloud-based parallel simulations were done [54,137].

The mOSAIC software platform is used by the prototypes of services delivering HPC-in-the-Cloud in the frame of HOST project [138] funded by European Commission in the frame of FP7 Capacities programme [139].

The mechanism used in Clouds and Grids for resource identification and brokering are close. Key ideas that were the basis for mOSAIC Ontology and Brokering systems were used recently in the Grid context. An Ontology for contract negotiations was presented in [140], while negotiations in an agent-based grid resource brokering system are exposed in [141].

An adaptive and semantic database model for RDF data stores [142] was also conceived following mOSAIC data service examples.

Related software products

Olaii [143] is a commercial product emerged as side effect of developing the dynamic semantic discovery service of mOSAIC. Information extraction library developed under the umbrella of mOSAIC's dynamic semantic discovery service is focused on extraction of the semantic descriptors for REST APIs, but can be extended to cover other use cases. The product is an application which will help discovering events or finding friends to go out with. The semantic extractor developed in mOSAIC is modified to extract the events instead of REST operations. Machine learning techniques applied in the semantic extractor to classify REST operations and to find irregular operations are applied for the events for building a recommendation system based on users' Facebook or Twitter profiles.

mOSAIC is currently used to provide an information service to the citizens of the third largest town of Romania. More precisely in the frame of SEED project [144], funded by European Commission through the CIP programme [145], a particular information service was build using mOSAIC API and platform and continuously extracts feeds from governmental sites (European, national and regional) as well as public service institutions (theaters, public transportation etc) and display them on in- and out-door large devices. The application is similar to the real-time-feed demo application.

mOSAIC positioning

Position as a solution for interoperability and portability in multiple Clouds usage scenarios

Generally speaking, the interoperability problem has three dimensions in the case of Cloud computing domain:

1. a design dimension, that deals with the need to abstract the programmatic diverse interfaces of various services,
2. a dynamic run-time dimension, that deals with the need to support migration of the Cloud application from one provider Cloud to another provider Cloud, and
3. a policy dimension, that deals with the need to support communication and federation among the Cloud providers.

mOSAIC is dealing mainly with the first dimension, ensuring an abstraction level for vendor-agnosticity. While is not tackling explicitly the migration or federation, it allows on-demand re-deployment of the supported application in various Clouds.

The portability problem has also three dimensions (all tackled by mOSAIC):

1. a functional dimension, that refers to the application functionality in an environment-agnostic manner,
2. a service dimension, that refers to the on-the-fly adding, reconfiguration and removal of resources, and
3. a data dimension, that refers to the import and export of data in different formats.

The main requirements of portability (following the comprehensive list from [146]) are met by the mOSAIC solution as indicated in Table 3.

There are currently several technical approaches to deal with portability and interoperability: open APIs and protocols (like jclouds [147], libcloud [148], OpenStack, OCCI [149] or δ-Cloud [150]), standards (like OVF [151], CDMI [152] or CIMI [153]), frameworks (like for SLAs from SLA@SOI project), semantic repositories (like UCI [154]), or domain specific languages (like CloudML [155]). mOSAIC is an *integrated solution* that offers an open API with a high level of abstraction, and uses OCCI, SLA@SOI framework and semantic processing.

Position as enabler for Multi-Cloud, Federations and InterCloud

There are several reasons for the use of services from multiple Clouds. The ones motivating mOSAIC are the following two: (1) ensure the avoidance of vendor lock-in by relaying upon the services from two or more providers; (2) support Hybrid Clouds build from Private and Public Clouds in order to deal with peaks or customer requirements.

According to the NIST report [156], multiple Clouds can be used sequentially or simultaneous. The sequentially usage is related to the migration from one Cloud to another driven from economic reasons (e.g. cost reductions, emergencies, back-ups etc). The simultaneous usage of services from different Clouds can also have several benefits like high availability and fault tolerance. mOSAIC is mainly targeting the first scenario, while is not excluding the second one.

According to [157], two delivery models can distinguished for multiple Clouds. The first one, the Federated

Table 3 Portability requirements (five most important ones in of the six category) and their degree of fulfillment by mOSAIC

Requirement	Fulfillment	Requirement	Fulfillment
Market		***Monitoring***	
Economic models	Model of a component-based market	SLA and performance monitoring	SLA compliance checks, component monitoring
License flexibility	Open-source	Sets of benchmarks	Benchmarks for component based applications
Negotiated SLAs	Through Cloud Agency	Load balance monitor	Through Containers
Cost-effectiveness	Broker mechanism	Service audit	—
Leasing mechanisms	—	QoS aware services	—
Application		***Deployment***	
Data portability and exchange	Unique API for same type of data services	Deploy in multiple Clouds with single tool	PaaS' Deployer
Scale-in and -out	Elasticity of Cloudlets	Service discovery	Based on semantics
Location-free	No location restriction	Automated provisioning	Requires user consensus
Workflow management	Components started in requested order	Navigation between services	—
Span on multiple Clouds	If no communication	Behavior prediction	—
Programming		***AA & Security***	
Minimal reimplementation when move	Re-deployment	Trust mechanisms	Intrusion-detection as a service
Common set or standard APIs	Use AMQP, OCCI	Authentication	Credential service
Same tools for cloud and entreprise appls	Eclipse and Web GUIs	Security standards	Use Cloud vendor certificates
Ontology of Cloud	Own Cloud Ontology	Single sign-on	—
High level modelling	—	Digital identities	—

Cloud, assumes a formal agreement between the Cloud providers. The second model, of Multi-Cloud, assumes that there is no priori agreement between the Cloud providers. mOSAIC is targeting mainly the Multi-Cloud, establishing only at deployment phase the needs in terms of services and the contacts with the Cloud providers.

According to [158], the term Multi-Cloud denotes the usage of multiple and independent Clouds by a client or a service. Clients or their software representatives are responsible for managing resource provisioning. In the same paper Multi-Clouds are classified in two categories. The first category is that of libraries allowing the usage of multiple clouds in a uniform way (including brokers that directly take care of provisioning of services across Clouds). The second category is that of services for provisioning resources which are hosted either externally or in-house by the clients, and which usually includes a broker. In this case the clients are entitled to specify a service level agreement or a set of provisioning rules and the service performs accordingly the deployment and execution. As the authors of [158] have correctly noted, mOSAIC can be mapped to the second category, of services.

We should remind that in the category of libraries, the most known products are the Python library Libcloud, the Java library jclouds, the REST API δ-Cloud, the PHP API Simple Cloud [159], or C++/ Python/Java API SAGA [160]. These libraries can be used as Drivers in mOSAIC. Moreover, mOSAIC's API libraries are available for Java, Python, Erlang and Node.js.

In what concerns the Multi-Clouds based on services, we consider that there are two categories: hosted services and deployable services. mOSAIC belongs to the deployable category.

Three hosted services are most relevant in this moment: RightScale, Enstratius and Kaavo. RightScale [161] offers a Private Cloud management platform for control, administration, and life-cycle support of deployments across multiple Clouds (it supports Amazon, Eucalyptus, GoGrid, VMware and FlexiScale); server templates are available to automatically install software on supported Cloud infrastructures. Enstratius [162] allows configuration management, monitoring, governance and automation and it supports Amazon, CloudStack, CloudSigma, Eucalyptus, GoGrid, Joyent Cloud, OpenStack, Rackspace, vCloud, Azure and others. Kaavo [163] allows also the deployment and management of distributed applications, workloads, and environments in various Clouds enabling resource management across Public, Private, and Hybrid Clouds (it supports Amazon, Rackspace, OpenStack, Eucalyptus, Cloud.com, Terremark, Logicworks, HP Cloud and IBM Cloud).

In terms of number of Cloud providers that are supported, mOSAIC is similar with the above mentioned hosted services, but with a clear preference for the support of European Cloud providers. In terms of GUIs the mOSAIC Web oriented one is considerably different, as designed to serve applications, not resource management: the user does not control the resources distribution or consumption (provider interfaces can be used for this purpose), but instead controls the processes of the application that are running (an application oriented view, instead a resource provider view).

The current deployable services for Multi-Cloud are in prototype phases as results of different research projects. Two main competitors for mOSAIC are currently available: Aoleus and Optimis. The RedHat's Aoleus [164] is an open-source Cloud management software written in Ruby for Linux systems. It allows users to choose between Private, Public or Hybrid Clouds, using δ-Cloud library. The Optimis Toolkit [165], result of the project with the same name and funded by the European Commission in the same work-programme as mOSAIC (in parallel with it), offers a platform for Cloud service provisioning that manages the lifecycle of the service and addresses issues like risk and trust management. Compared with Aoleus, mOSAIC has the advantage of including more than a resource management middleware (not restricted to the δ-Cloud list of providers). Compared with Optimis, mOSAIC is weaker in terms of trust and risk management, or even in brokerage process due to the reduced set of policies; however the semantic processing support, the SLA-based negotiation mechanisms, or the application tools are the main comparative advantages of mOSAIC.

A comprehensive analysis of mOSAIC positioning versus the research prototypes for Federations and Multi-Clouds, complementary to the above one that is mentioning software products, was recently exposed in [166].

A Cloud Federation or a Multi-Cloud that includes at least one Cloud Broker and offers dynamic service provisioning is an Inter-Cloud. In the case of Multi-Cloud, the Broker is often part of the service or library. This is the case also for mOSAIC. According [158], the brokering mechanisms are: SLA-based, i.e. requirements are specified by clients in the form of a service level agreements; or trigger-action, i.e. rules are becoming active, triggering an action, when a predefined condition considering the externally visible application performance indicators becomes true. Evidently, mOSAIC has a SLA-based brokering mechanism. However, taking into account its event-driven orientation, part of its behavior is based on rules and triggered-actions.

The most relevant representatives for SLA-based brokers for Multi-Clouds are SpotCloud and Stratos. SpotCloud [167] provides a marketplace where service providers sell the extra capacity they have and the clients can select the 'best' service provider at a certain moment. WSO2 Stratos [168] provides core services and building blocks for federated identity and single sign-on, data-as-a-service and messaging-as-a-service, and monitors SLAs, CPU, memory and bandwidth utilization and automatically scales up or down depending on the load. The mOSAIC brokering, monitoring, scaling or messaging mechanisms are rather simpler compared with SpotCloud and Stratos ones. However, its component-based and open-source design has the comparative advantage of easy updates and customization according to user's needs.

mOSAIC has not been yet able to establish a market place, as SpotCloud, but it does not exclude the idea. A Cloud platform supporting applications composed of software components can come with a component store which provides components for common tasks. Using such a store, developers should be able to build new applications and services by configure and compose the existing components, or by extending them with new functionalities. Moreover, a dynamic recomposition of software during execution, i.e. adding, removing or reconfiguring components within an application at runtime, should be possible. A case study of using mOSAIC and a component market for the design of a Bussiness-Process-as-a-Service is sketched in [169].

Position as PaaS

As described earlier, mOSAIC exhibits some characteristics of a PaaS. Despite the fact that is not a hosted service, but a deployable one, it offers several tools and facilities to develop, deploy and control at run-time new applications. We consider here only two significant PaaSs to be compared with mOSAIC's PaaS. While the PaaS offer is quite diverse, the selection is based on the usage spread, respectively closeness to mOSAIC.

Google's Application Engine (GAE) is one of the most known PaaS. While the deployment facilities are more complicated in mOSAIC due to the need of the user final decision in selecting the Cloud provider, several limitations of GAE are avoided by mOSAIC. For example filesystem write access are forbidden in GAE; applications are not allowed to make arbitrary network connections to the Internet and HTTP requests must be made only through a special library. All data handled by GAE must be stored in a columnar database, and even though the developer has a query language resembling SQL, it is very limited in what concerns filters. Moreover, in GAE the requests should be handled within one minute or less. All of these restrictions are not encountered in mOSAIC approach.

From the long list of hosting PaaS, Heroku [170] is the most closest to mOSAIC concepts. Its features have inspired not only mOSAIC, but also most of the recently emerged PaaSs. The reason is its simple scheme to handle development, configuration, deployment, and management. It supports a variety of programming languages (Ruby, Python, Node.js, Scala and Clojure), as well as arbitrary executables either as binary or scripts. However, there are several improvements that mOSAIC has been able to provide. For example, it is actually impractical to mix multiple languages in the same Heroku application, while in mOSAIC components can be written in various languages if they are able to use a message passing system. Any update of component of a Heroku application needs the complete shutdown of the application. In mOSAIC components can be stopped, started or updated during the execution of the application, as the messages are waiting in the queues. Several other advantages are comprehensively discussed in [171].

Position in open-source community

As earlier mentioned, mOSAIC is included in the category of services for Multi-Cloud. However there are other open-source middlewares that are deployable, while not necessary designed with the Multi-Cloud in mind, still able to deal with homogeneous distributed resources. A comprehensive comparisons of them and mOSAIC positioning is presented in [172], including ConPaaS, the Contrail [173] solution for Federations. In the same paper are provided a series of criteria that can be used to compare PaaSs.

We mention here only VMware's Cloud Foundry and Red Hat's OpenShift [174], developed in the same time as mOSAIC. Both are dedicated to Web applications, while mOSAIC scope is more broader. mOSAIC is stronger also in terms of data support, the number of Cloud providers that are supported, the interfacing variants, the SLA and brokerage mechanisms, as well as portability on other Linux system than VMware or RedHat provided ones. However is weaker in terms of performance analytics or integration with other development environments than Eclipse.

Position in the landscape of Cloud Computing projects funded by the European Commission

Beyond Optimis, Contrail and MODAClouds that were mentioned earlier, there are several other Cloud computing projects that have run in parallel with mOSAIC and have provided close related solutions. A snapshot of the landscape covered by these projects in the late 2011 was provided with mOSAIC contribution in [175] in the book [176] that collects reports on the states of more than twelve such projects. The positioning of mOSAIC in relationship with several Multi-Cloud projects was expressed more recently in [177].

We remind here only few projects offering alternatives to mOSAIC approach or complementing it. 4CaaSt [178] is building a BluePrint for registering Cloud services in an e-Market. Cloud4SOA [179] is dealing with semantic based interoperability at platform level. Cloud-TM [180] is proposing another programming paradigm for Clouds. Remics [181] is dealing with migration of legacy applications to Clouds through model-driven engineering. TClouds [182] is offering security, privacy and resilience mechanisms for Multi-Clouds. Vision Cloud [183] is looking in details to the issues of data management in Federations and Multi-Clouds.

Conclusions and future developments

Drawing the line at the end of project, the mOSAIC's multi-national team is checking the degree of fulfillment of the initial promises. Shortly these were:

(i) a set of APIs for application portability between Clouds,
(ii) agent technologies supporting dynamic negotiations with multiple Cloud providers,
(iii) user-centric service level agreements,
(iv) Cloud ontology and semantic data processing,
(v) an open-source and portable platform-as-a-service,
(vi) proof-of-concept applications.

Targeting to provide an innovative solution in these fields, mOSAIC approach has proven its uniqueness and advantages over other existing approaches in what concerns:

(a) deployable and portable services of platform type on top of IaaS;
(b) brokering system based on customizable service level agreements and agent technologies;
(c) portability of Cloud applications supported by semantic processors and multi-layered API;
(d) usefulness for porting scientific and commercial applications towards the Cloud;
(e) a stable, complete and innovative middleware for building, deploying and controlling applications following Multi-Cloud usage scenarios.

While disparate proofs of the innovations are dispersed in various mOSAIC-related articles, the present report tried to offer a general overview of the main achievements and advantages of mOSAIC. However it highlights also some weaknesses in relationship with other approaches, subject to improvements in the next years:

(1) the event-driven programming style of mOSAIC applications is considered to be complex by less skilled programmers; the templates collections as well as the workbench and wizzards should be improved to better assist the application developers;
(2) the deployment of special libraries still require manual intervention; the deployment procedures are therefore expected to be improved;
(3) the mechanism of the brokerage system allow complex policies to be applied; however, simple policies are currently used and the full potential was not yet exploited;
(4) semantic processing is used currently at design phase; the potential of the dynamic discovery services, for example, has not yet been fully exploited.
(5) several software prototypes have been developed as proof-of-concept and a considerable part of them, yet functional, are not ready for a production phase; according to the interest expressed by the community surrounding mOSAIC, particular components (like the PTC simulator and resource allocator) will be further improved to offer production-level quality of service;
(6) commercial products developments has underline the need of enlarging the number of COTS that are wrapped to work with mOSAIC platform;
(7) the proof-of-concept applications developed in the frame of the project are expected to be improved to satisfy the requirements of their external users.

Several topics that were not in the main focus of mOSAIC project are expected to be pursued in the near future relying on the mOSAIC approach and software repositories. We have already mentioned some of them in this paper: model-driven engineering for Clouds, Cloud security or automated management of Cloud resources. With certainty this will happen in the frame of the research, development and collaborative projects that already rely upon the mOSAIC's specific components. We remind here some of them:

FP7 projects: MODAClouds in model-driven engineering direction, SPECS in security direction, HOST in scientific application support direction, Helix Nebula in Earth observation application field;
National projects: AMICAS, in the direction of automated management of multiple Cloud resources, or Cloud@Home in the direction of Volunteer Cloud.

The fact that the mOSAIC architecture is built from loosely coupled components enhance the chances for the open-source software prototypes to be adopted and enhanced in other contexts that mOSAIC initial scenarios. This is the case of the commercial product Olaii that was mentioned in this paper, which has started from the semantic extractor developed in the frame of mOSAIC project.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
The paper presents a comprehensive overview of the mOSAIC approach for Multi-Cloud. It offers the first integrated image of the full set of solutions, at the end of the development cycle. It also position the approach versus other currently available ones, from multiple points of view (Multi-Cloud, open-source, PaaS, related initiatives). All authors read and approved the final manuscript.

Acknowledgements
The research and development reported in this article was partially supported by the European Commission grant no. FP7-ICT-2009–5-256910 (mOSAIC). The development of the mOSAIC approach was sustained by a large multinational team that is far from being reflected in the list of the authors of this paper: the paper's authors are the key persons from each team that have drive the approach development. They are taking this opportunity to express their gratitude to the team members of mOSAIC project that are not in the author list:

– from Second University of Naples: Alba Amato, Giuseppe Aversano, Rocco Aversa, Antonio Bagarolo, Daniele Bove, Pasquale Cantiello, Giuseppina Cretella, Vincenzo De Crescenzo, Massimiliano Diodati, Massimo Ficco, Loredana Liccardo, Francesco Moscato, Manuela Serrao, Luca Tasquier;
– from Institute e-Austria Timişoara: Bogdan Alexandru Căprărescu, Adrian Copie, Ciprian Dorin Crăciun, Teodor-Florin Fortiş, Marc Eduard Frîncu, Georgiana Macariu, Victor Ion Munteanu, Marian Neagul, Viorel Negru, Silviu Panica, Călin Şandru;
– from AITIA: Rajmund Bocsi, Gabor Ferschl, László Gulyás;
– from Tecnalia: Unai Antero, Imanol Padillo Cruz, Gorka Mikel Echevarria Velez, Iñigo Lazkanotegui Larrate, Miguel Loichate, Asier Sesma;
– from TerraDue: Hervé Caumont, Pedro Gonçalves, Simone Tripodi;
– from ESRIN-ESA: Claudio Di Giulio, Fulvio Marelli;
– from Xlab: Mariano Cecowski, Gregor Pipan, Boris Savič;
– from Brno University of Technology: Petr Škoda, Pavel Smrž;
– from University of Ljubljana: Peter Češarek, Jernej Južna, Matija König, Aleš Kroflič, Gašper Stegnar.

Author details
[1]Institute e-Austria Timişoara and West University of Timişoara, Timişoara, Romania. [2]Second University of Naples, Aversa, Italy. [3]AITIA International Inc, Budapest, Hungary. [4]Industrial Systems Unit, Tecnalia, San Sebastian, Spain. [5]Terradue SRL, Rome, Italy. [6]Earth Observation Science, Applications and Future Technologies Department, ESRIN, European Space Agency, Frascati, Italy. [7]XLAB d.o.o., Ljubljana, Slovenia. [8]Brno University of Technology, Brno, Czech Republic. [9]University of Ljubljana, Ljubljana, Slovenia.

Received: 12 January 2013 Accepted: 7 May 2013
Published: 24 May 2013

References
1. Armbrust M, Fox A, Griffith R, Joseph AD, Katz RH, Konwinski A, Lee G, Patterson DA, Rabkin A, Zaharia M (2009) Above the clouds: A berkeley view of cloud computing. Tech. rep., EECS Department, U.C. Berkley, UCB/EECS-2009–28. http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf
2. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I (2009) Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Comput Syst 25(6): 599–616. doi:10.1016/j.future.2008.12.001

3. Schubert L, Jeffery Ke (2012) Above the clouds: A berkeley view of cloud computing. Tech. rep., European Commission, Expert Group on Cloud Computing. http://cordis.europa.eu/fp7/ict/ssai/docs/future-cc-2may-finalreport-experts.pdf
4. Ict – information and communication technologies. http://cordis.europa.eu/fp7/ict/
5. mOSAIC project. http://www.mosaic-cloud.eu
6. Di Martino B, Petcu D, Cossu R, Gonçalves P, Máhr T, Loichate M (2011b) Building a mosaic of clouds. In: Guarracino MR, Vivien F, Täff JL, Cannatoro M, Danelutto M, Hast A, Perla F, Knüpfer A, Martino B, Alexander M (eds) Euro-Par 2010 Parallel Processing Workshops, Lecture Notes in Computer Science, vol 6586. Springer, pp 571–578. doi:10.1007/978-3-642-21878-1_70, http://dx.doi.org/10.1007/978-3-642-21878-1_70
7. Malawski M, Meizner J, Bubak M, Gepner P (2011) Component approach to computational applications on clouds. Procedia Comput Sci 4(0): 432–441. doi:10.1016/j.procs.2011.04.045, http://dx.doi.org/10.1016/j.procs.2011.04.045
8. Petcu D, Şandru C (2012) Towards component-based software engineering of cloud applications In: WICSA/ECSA 2012 Companion Volume. ACM, pp 80–81. doi:10.1145/2361999.2362013, http://doi.acm.org/10.1145/2361999.2362013
9. Kächele S, Domaschka J, Hauck FJ (2011) Cosca: an easy-to-use component-based paas cloud system for common applications In: 1st International Workshop on Cloud Computing Platforms (CloudCP 2011). ACM, pp 4:1–4:6. doi:10.1145/1967422.1967426, http://doi.acm.org/10.1145/1967422.1967426
10. Advanced message queuing protocol. http://www.amqp.org/
11. Amazon simple queue service. http://aws.amazon.com/sqs/
12. Petcu D, Crăciun C, Rak M (2011c) Towards a cross platform cloud api - components for cloud federation. In: Leymann F, Ivanov I, van Sinderen M, Shishkov B (eds) 1st International Conference on Cloud Computing and Services Science (CLOSER 2011). SciTePress, pp 166–169
13. Petcu D, Macariu G, Panica S, Crăciun C (2012b) Portable cloud applications—from theory to practice. Future Generation Comput Syst 29(6): 1417–1430. doi:10.1016/j.future.2012.01.009, http://dx.doi.org/10.1016/j.future.2012.01.009
14. Petcu D, Panica S, Şandru C, Crăciun CD, Neagul M (2012c) Experiences in building an event-driven and deployable platform as a service. In: Wang XS, Cruz I, Delis A, Huang G (eds) Web Information Systems Engineering (WISE 2012), Lecture Notes in Computer Science, vol 7651. Springer, pp 666–672. doi:10.1007/978-3-642-35063-4_51, http://dx.doi.org/10.1007/978-3-642-35063-4_51
15. Petcu D, Crăciun C, Neagul M, Lazcanotegui I, Rak M (2011a) Building an interoperability api for sky computing In: International Conference on High Performance Computing and Simulation (HPCS 2011). IEEE Computer Society Press, pp 405–411. http://dx.doi.org/10.1109/HPCSim.2011.5999853
16. Cloud Computing Use Case Discussion Group (2010) Cloud computing use cases, version 4.0. Tech. rep. http://opencloudmanifesto.org/Cloud_Computing_Use_Cases_Whitepaper-4_0.pdf
17. Petcu D (2010) Identifying cloud computing usage patterns In: IEEE International Conference on Cluster Computing Workshops and Posters (Cluster Workshops 2010). IEEE Computer Society Press, pp 1–8. http://dx.doi.org/10.1109/CLUSTERWKSP.2010.5613106
18. Fortiş TF, Lopez GE, Cruz IP, Ferschl G, Máhr T (2012b) Cloud patterns for mosaic-enabled scientific applications In: International Conference on Parallel Processing (Euro-Par 2011). Springer, Lecture Notes in Computer Science, vol 7156, pp 83–93. doi:10.1007/978-3-642-29737-3_10, http://dx.doi.org/10.1007/978-3-642-29737-3_10
19. Aversa R, Di Martino B, Rak M, Venticinque S (2010) Cloud agency: A mobile agent based cloud system In: International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2010). IEEE Computer Society Press, pp 132–137. http://dx.doi.org/10.1109/CISIS.2010.143
20. Şandru C, Venticinque S (2013) Agents layer to support cloud applications. In: Fortino G, Badica C, Malgeri M, Unland R (eds) Intelligent Distributed Computing VI, Studies in Computational Intelligence, vol 446. Springer, pp 281–286. doi:10.1007/978-3-642-32524-3_35, http://dx.doi.org/10.1007/978-3-642-32524-3_35
21. Cuomo A, Rak M, Venticinque S, Villano U (2012) Enhancing an autonomic cloud architecture with mobile agents In: International Conference on Parallel Processing (Euro-Par 2011). Springer, Lecture Notes in Computer Science, vol 7156, pp 94–103. doi:10.1007/978-3-642-29737-3_11, http://dx.doi.org/10.1007/978-3-642-29737-3_11
22. Venticinque S, Aversa R, Di Martino B, Petcu D (2011a) Agent based cloud provisioning and management - design and prototypal implementation. In: Leymann F, Ivanov I, v Sinderen M, Shishkov B (eds) 1st International Conference on Cloud Computing & Services Science (CLOSER 2011). SciTePress, pp 184–191
23. Tasquier L, Venticinque S, Aversa R, Di Martino B (2012) Agent based application tools for cloud provisioning and management. In: Yousif M, Schubert L, Jeffery K (eds) 3rd International Conference on Cloud Computing (CloudComp 2012). pp 24–30. http://www.itutility.ac.uk/outputs/CloudComp_Wien_Sept2012/USB-cloudcomp2012/file-storage/papers/134555472397963.pdf
24. Venticinque S, Amato A, Di Martino B (2012a) An occi compliant interface for iaas provisioning and monitoring. In: Leymann F, Ivanov I, van Sinderen M, Shan T (eds) 2nd International Conference on Cloud Computing and Services Science (CLOSER 2012). SciTePress, pp 163–166
25. Cao BQ, Li B, Xia QM (2009) A service-oriented qos-assured and multi-agent cloud computing architecture In: 1st International Conference on Cloud Computing (CloudCom 2009). Springer-Verlag, pp 644–649. doi:10.1007/978-3-642-10665-1_66, http://dx.doi.org/10.1007/978-3-642-10665-1_66
26. Venticinque S, Tasquier L, Di Martino B (2012c) Agents based cloud computing interface for resource provisioning and management In: 6th International Conference on Complex, Intelligent & Software Intensive Systems (CISIS 2012). IEEE Computer Society, pp 249–256. http://dx.doi.org/10.1109/CISIS.2012.139
27. Amato A, Venticinque S, Di Martino B (2012b) Evaluation and brokering of service level agreements for negotiation of cloud infrastructures In: International Conference for Internet Technology and Secured Transactions (ICITST-2012). Infonomics Society, pp 144–149
28. Aversa R, Tasquier L, Venticinque S (2012) Management of cloud infrastructures through agents In: 3rd International Conference on Emerging Intelligent Data and Web Technologies (EIDWT 2012). IEEE Computer Society Press, pp 46–53. http://dx.doi.org/10.1109/EIDWT.2012.57
29. Ficco M, Venticinque S, Di Martino B (2012) Mosaic-based intrusion detection framework for cloud computing. In: Meersman R, Panetto H, Dillon T, Rinderle-Ma S, Dadam P, Zhou X, Pearson S, Ferscha A, Bergamaschi S, Cruz I (eds) On the Move to Meaningful Internet Systems: OTM 2012. Springer-Verlag, Lecture Notes in Computer Science, vol 7566, pp 628–644.
30. Amato A, Tasquier L, Copie A (2013b) Vendor agents for iaas cloud interoperability. In: Fortino G, Badica C, Malgeri M, Unland R (eds) Intelligent Distributed Computing VI Studies in Computational Intelligence, vol 446. Springer, pp 271–280. doi:10.1007/978-3-642-32524-3_34, http://dx.doi.org/10.1007/978-3-642-32524-3_34
31. Şandru C, Petcu D, Munteanu V (2012) Building an open-source platform-as-a-service with intelligent management of multiple cloud resources In: 5th IEEE/ACM Internat. Conference on Utility & Cloud Computing (UCC 2012). IEEE Computer Society, pp 333–338. http://dx.doi.org/10.1109/UCC.2012.54
32. Rackspace. http://www.rackspace.com
33. GoGrid. http://www.gogrid.com
34. Flexiant. http://www.flexiant.com
35. CloudSigma. http://www.cloudsigma.com
36. NIIFI Cloud. http://www.cloudsigma.com
37. Arctur vCloud. https://vcloud.arctur.si/cloud/
38. Hostko. http://www.hostko.si
39. VMware vCloud. http://vcloud.vmware.com
40. OnApp. http://onapp.com
41. Eucalyptus. http://www.eucalyptus.com
42. OpenNebula. http://opennebula.org
43. CloudStack. http://cloudstack.apache.org
44. OpenStack. http://www.openstack.org/
45. SLA@SOI project. http://sla-at-soi.eu/

46. Theilmann W, Yahyapour R, Butler J (2008) Multi-level sla management for service-oriented infrastructures. In: Mähönen P, Pohl K, Priol T (eds) Towards a Service-Based Internet, Springer Berlin, Lecture Notes in Computer Science, vol 5377, pp 324–335. http://dx.doi.org/10.1007/978-3-540-89897-9_28
47. Comuzzi M, Kotsokalis C, Rathfelder C, Theilmann W, Winkler U, Zacco G (2010) A framework for multi-level sla management. In: Dan A, Gittler F, Toumani F (eds) ICSOC/ServiceWave 2009 Workshops, Springer Berlin. Lecture Notes in Computer Science, vol 6275, pp 187–196. http://dx.doi.org/10.1007/978-3-642-16132-2_18
48. WS-Agreement for Java framework. http://wsag4j.sourceforge.net
49. Rak M, Aversa R, Venticinque S, Di Martino B (2012a) User centric service level management in mosaic applications. In: Alexander M, D'Ambra P, Belloum A, Bosilca G, Cannataro M, Danelutto M, Di Martino B, Gerndt M, Jeannot E, Namyst R, Roman J, Scott S, Traff J, Vallée G, Weidendorfer J (eds) Euro-Par 2011: Parallel Processing Workshops, Lecture Notes in Computer Science, vol 7156, Springer, pp 106–115. doi:10.1007/978-3-642-29740-3_13, http://dx.doi.org/10.1007/978-3-642-29740-3_13
50. Ficco M, Rak M (2011) Intrusion tolerant approach for denial of service attacks to web services In: International Conference on Data Compression, Communications and Processing (CCP 2011). IEEE Computer Society Press, Los Alamitos, pp 285–292. http://dx.doi.org/10.1109/CCP.2011.44
51. Ficco M, Rak M (2012a) Intrusion tolerance in cloud applications: The mosaic approach In: 6th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2012). IEEE Computer Society Press, pp 170–176. http://dx.doi.org/10.1109/CISIS.2012.202
52. Ficco M, Rak M (2012b) Intrusion tolerance of stealth dos attacks to web services. In: Gritzalis D, Furnell S, Theoharidou M (eds) SEC 2012, Springer, IFIP Advances in Information and Communication Technology, vol 376, pp 579–584. http://dx.doi.org/10.1007/978-3-642-30436-1_52
53. Rak M, Liccardo L, Aversa R (2011b) A sla-based interface for security management in cloud and grid integrations In: 7th International Conference on Information Assurance and Security (IAS 2011). IEEE Computer Society Press, pp 378–383. http://dx.doi.org/10.1109/ISIAS.2011.6122783
54. Rak M, Cuomo A, Villano U (2012b) Cloud-based concurrent simulation at work: Fast performance prediction of parallel programs In: 21st IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2012). IEEE Computer Society Press, pp 137–142. http://dx.doi.org/10.1109/WETICE.2012.74
55. Rak M, Cuomo A, Villano U (2012c) Mjades: Concurrent simulation in the cloud. In: Barolli L, Xhafa F, Vitabile S, Uehara M (eds) 6th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2012). IEEE Computer Society Press, pp 853–860. http://dx.doi.org/10.1109/CISIS.2012.134
56. Amato A, Venticinque S (2013) Multi-objective decision support for brokering of cloud sla In: 27th IEEE International Conference on Advanced Information Networking and Applications Workshops (WAINA 2013). IEEE Computer Society, pp 1241–1246. http://dx.doi.org/10.1109/WAINA.2013.149
57. Venticinque S, Negru V, Munteanu VI, Şandru C, Aversa R, Rak M (2012b) Negotiation policies for provisioning of cloud resources. In: Filipe J, Fred ALN (eds) 4th International Conference on Agents and Artificial Intelligence (ICAART 2012), SciTePress, pp 347–350
58. Amato A, Liccardo L, Rak M, Venticinque S (2012a) Sla negotiation and brokering for sky computing. In: Leymann F, Ivanov I, van Sinderen M, Shan T (eds) 2nd International Conference on Cloud Computing and Services Science (CLOSER 2012). SciTePress, pp 611–620
59. Venticinque S (2012) Agent based services for negotiation, monitoring and reconfiguration of cloud resources In: European Research Activities in Cloud Computing. Cambridge Scholars Publishing, pp 178–202. http://www.c-s-p.org/Flyers/European-Research-Activities-in-Cloud-Computing1-4438-3507-2.htm
60. Venticinque S, Aversa R, Di Martino B, Rak M, Petcu D (2011b) A cloud agency for sla negotiation and management. In: Guarracino MR, Vivien F, Träff JL, Cannatoro M, Danelutto M, Hast A, Perla F, Knüpfer A, Di Martino B, Alexander M (eds) Euro-Par 2010 Parallel Processing Workshops, Lecture Notes in Computer Science, vol 6586. Springer, pp 587–594. doi:10.1007/978-3-642-21878-1_72, http://dx.doi.org/10.1007/978-3-642-21878-1_72
61. Cretella G, Di Martino B (2012a) Towards a semantic engine for cloud applications development In: 6th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2012). IEEE Computer Society Press, pp 198–203. http://dx.doi.org/10.1109/CISIS.2012.159
62. Cretella G, Di Martino B (2012) Semantic web annotation and representation of cloud apis In: 3rd International Conference on Emerging Intelligent Data and Web Technologies (EIDWT 2012). IEEE Computer Society Press, pp 31–37. http://dx.doi.org/10.1109/EIDWT.2012.61
63. Cretella G, Di Martino B (2012b) Towards automatic analysis of cloud vendors apis for supporting cloud application portability In: 6th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2012). IEEE Computer Society Press, pp 61–67. http://dx.doi.org/10.1109/CISIS.2012.162
64. Di Martino B, Cretella G (2012) Semantic and algorithmic recognition support to porting software applications to cloud. In: Bibi S, Moschitti A, Plank B, Stamelos I (eds) Joint Workshop on Intelligent Methods for Software System Engineering (JIMSE 2012), pp 24–30. http://www2.lirmm.fr/ecai2012/images/stories/ecai_doc/pdf/workshop/W26-jimse2012.pdf
65. Amato A, Cretella G, Di Martino B, Venticinque S (2013a) Semantic and agent technologies for cloud vendor agnostic resource brokering In: 27th International Conference on Advanced Information Networking and Applications Workshops (WAINA 2013). IEEE Computer Society, pp 1253–1258. http://dx.doi.org/10.1109/WAINA.2013.163, http://www.odysci.com/article/1010113019663301/semantic-and-agent-technologies-for-cloud-vendor-agnostic-resource-brokering
66. Youseff L, Butrico M, Da Silva D (2008) Toward a unified ontology of cloud computing In: Grid Computing Environments Workshop (GCE 2008), pp 1–10. doi:10.1109/GCE.2008.4738443, http://dx.doi.org/10.1109/GCE.2008.4738443
67. Moscato F, Aversa R, Di Martino B, Fortiş T, Munteanu V (2011) An analysis of mosaic ontology for cloud resources annotation In: Federated Conference on Computer Science and Information Systems (FedCSIS 2011), pp 973–980. http://fedcsis.eucip.pl/proceedings/pliks/154.pdf
68. Aversa R, Di Martino B, Moscato F, Petcu D, Rak M, Venticinque S (2011b) An ontology for the cloud in mosaic. In: Wang L, Ranjan R, Chen J, Benatallah B (eds) Cloud Computing: Methodology Systems, and Applications. CRC Press, pp 467–486. http://www.crcpress.com/product/isbn/9781439856413
69. Fortis TF, Munteanu VI, Negru V (2012) Towards an ontology for cloud services In: 6th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS 2012). IEEE Computer Society Press, Washington, pp 787–792. doi:10.1109/CISIS.2012.138, http://dx.doi.org/10.1109/CISIS.2012.138
70. RabbitMQ. http://www.rabbitmq.com/
71. Amazon Simple Storage Service. http://aws.amazon.com/s3/
72. Riak. http://basho.com/riak/
73. Hadoop Distributed File System. http://hadoop.apache.org/docs/r1.0.4/hdfs_design.html
74. Petcu D, Crăciun C, Neagul M, Panica S, Di Martino B, Venticinque S, Rak M, Aversa R (2011b) Architecturing a sky computing platform. In: Cezon M, Wolfsthal Y (eds) Towards a Service-Based Internet. ServiceWave 2010 Workshops, Lecture Notes in Computer Science, vol 6569. Springer, pp 1–13. doi:10.1007/978-3-642-22760-8_1, http://dx.doi.org/10.1007/978-3-642-22760-8_1
75. Petcu D (2012b) How to build a reliable mosaic of multiple cloud services In: 1st European Workshop on Dependable Cloud Computing (EWDCC 2012). ACM, pp 4:1–4:2. doi:10.1145/2365316.2365320, http://doi.acm.org/10.1145/2365316.2365320
76. CouchDB. http://couchdb.apache.org/
77. Jetty. http://www.eclipse.org/jetty/
78. VirtualBox. https://www.virtualbox.org/
79. mOSAIC's open-source code repository. https://bitbucket.org/mosaic
80. Google App Engine. https://appengine.google.com
81. Windows Azure. http://www.windowsazure.com
82. Cloud Foundry. http://www.cloudfoundry.com
83. Petcu D, Frîncu ME, Crăciun C, Panica S, Neagul M, Macariu G (2011d) Towards open-source cloudware In: 4th IEEE International Conference

on Utility and Cloud Computing (UCC 2011). IEEE Computer Society Press, pp 330–331. http://dx.doi.org/10.1109/UCC.2011.53

84. Cossu R, Di Giulio C, Brito F, Petcu D (2013) Cloud computing for earth observation. In: Kyriazis D, Voulodimos A, Gogouvitis S, Varvarigou T (eds) Data Intensive Storage Services for Cloud Environments, IGI Global, chap 12, pp 256–277. http://dx.doi.org/10.4018/978-1-4666-3934-8
85. Panica S, Petcu D, Lazkanotegi Larrate I, Máhr T (2012) Sky computing platform for legacy distributed application In: International Symposium on Parallel and Distributed Computing (ISPDC 2012). IEEE Computer Society Press, pp 293–300. http://dx.doi.org/10.1109/ISPDC.2012.47
86. Ferschl G, Máhr T (2012) Migrating a simulation framework from the cloud to the sky In: WoSS-4, CLASS Conference, Slovenia, pp 25–27
87. Škoda P, Šperka S, Smrž P (2012) Extracting information from scientific papers in the cloud In: 6th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2012). IEEE Computer Society Press, pp 775–780. http://dx.doi.org/10.1109/CISIS.2012.176
88. Šperka S, Škoda P, Smrž P (2012) Cloudification of legacy information extraction system In: WoSS-4, CLASS Conference, Slovenia, pp 9–14. http://www.kc-class.eu/datoteke/proceedings-woss-4.pdf
89. Stankovski V, König M (2012) A sustainable building application design based on the mosaic api and platform In: 8th International Conference on Semantics, Knowledge and Grid (SKG 2012), pp 249–252. doi:10.1109/SKG.2012.13, http://dx.doi.org/10.1109/SKG.2012.13
90. Južna J, Češarek P, Stankovski V (2013) Porting existing matlab applications to the cloud by using the mosaic platform. In: Topping BHV, Iványi P (eds) 3rd International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering (PARENG 2013). Civil-Comp Press. paper 30. http://dx.doi.org/10.4203/ccp.101.30
91. Cretella G, Di Martino B, Stankovski V (2012) Using the mosaic's semantic engine to design and develop civil engineering cloud applications. In: Taniar D, Pardede E, Steinbauer M, Khalil I (eds) 14th International Conference on Information Integration and Web-based Applications & Services (iiWAS2012). ACM, pp 378–386. http://dx.doi.org/10.1145/2428736.2428805
92. Rak M, Aversano G (2012) Benchmarks in the cloud: The mosaic benchmarking framework In: 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2012). IEEE Computer Society, pp 415–422. http://dx.doi.org/10.1109/SYNASC.2012.41
93. Cuomo A, Rak M, Villano U (2013b) Simulation-based performance evaluation of cloud applications. In: Fortino G, Badica C, Malgeri M, Unland R (eds) Intelligent Distributed Computing VI Studies in Computational Intelligence, vol 446. Springer, pp 263–269. doi:10.1007/978-3-642-32524-3_33, http://dx.doi.org/10.1007/978-3-642-32524-3_33
94. Petcu D (2012a) Cloudware support for scientific applications In: RO-LCG 2012. IEEE, Romania, pp 70–73
95. Petcu D, Panica S, Neagul M (2010) From grid computing towards sky computing. case study for earth observation In: 10th Cracow Grid Workshop (CGW 2010). Academic Computer Center, Poland, pp 11–20
96. Panica S, Neagul M, Crăciun C, Petcu D (2011) Serving legacy distributed applications by a self-configuring cloud processing platform In: 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS 2011), vol 1. IEEE Computer Society Press, pp 139–144. http://dx.doi.org/10.1109/IDAACS.2011.6072727
97. Stankovski V, Južna J, Petcu D (2012) Enabling legacy engineering applications for cloud computing: Experience with the mosaic api and platform In: 3rd International Conference on Emerging Intelligent Data and Web Technologies (EIDWT 2012). IEEE Computer Society Press, pp 281–286. http://dx.doi.org/10.1109/EIDWT.2012.49
98. Rak M, Ficco M, Luna J, Ghani H, Suri N, Panica S, Petcu D (2012d) Security issues in cloud federation. In: Villari M, Brandic I, Tusa F (eds) Achieving Federated and Self-Manageable Cloud Infrastructures: Theory and Practice. IGI Global, pp 176–194. http://dx.doi.org/10.4018/978-1-4666-1631-8.ch010
99. Casola V, Cuomo A, Rak M, Villano U (2013) The cloudgrid approach: Security analysis and performance evaluation. Future Generation Comput Syst 29(1): 387–401. doi:10.1016/j.future.2011.08.008, http://dx.doi.org/10.1016/j.future.2011.08.008
100. Casola V, Cuomo A, Villano U, Rak M (2012) Access control in federated clouds: The cloudgrid case study. In: Villari M, Brandic I, Tusa F (eds) Achieving Federated and Self-Manageable Cloud Infrastructures: Theory and Practice. IGI Global, pp 395–417. doi:10.4018/978-1-4666-1631-8.ch020, http://dx.doi.org/10.4018/978-1-4666-1631-8.ch020
101. SPECS project. http://www.fp7-specs.eu
102. Rak M, Ficco M (2012) Intrusion tolerance as a service - a sla-based solution. In: Leymann F, Ivanov I, van Sinderen M, Shan T (eds) 2nd International Conference on Cloud Computing and Services Science (CLOSER 2012). SciTePress, pp 375–384
103. Rak M, Liccardo L, Aversa R (2012e) A sla-based interface for secure authentication negotiation in cloud. J Inf Assur Secur 7(2): 137–146. http://www.mirlabs.org/jias/secured/Volume7-Issue3/vol7-issue3.html
104. Bennett T, Stephen an Erl, Gee C, Laird R, Manes AT, Schneider R, Shuster L, Tost A, Venable C (2011) SOA Governance: Governing Shared Services On-Premise & in the Cloud. Prentice Hall/Pearson PTR
105. Munteanu VI, Fortiş T F, Negru V (2012b) An event driven multi-agent architecture for enabling cloud governance. IEEE Computer Society Press. http://dx.doi.org/10.1109/UCC.2012.50
106. Fortiş T, Munteanu V, Negru V (2012a) Steps towards cloud governance. a survey In: 34th Internat. Conference on Information Technology Interfaces (ITI 2012), pp 29–34. http://dx.doi.org/10.2498/iti.2012.0374
107. Copie A, Fortis TF, Munteanu VI, Negru V (2012b) Datastores supporting services lifecycle in the framework of cloud governance. Scalable Comput: Pract Exp 13(3): 251–267. https://www.scpe.org/index.php/scpe/article/view/796
108. Munteanu VI, Fortiş TF, Copie A (2012a) Building a cloud governance bus. Int J Comput Commun Control 7(5): 900–906. http://journal.univagora.ro/download/pdf/642.pdf
109. Copie A, Fortis T, Munteanu V, Negru V (2012a) Service datastores in cloud governance In: 10th IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA 2012). IEEE Computer Society Press, pp 473–478. http://dx.doi.org/10.1109/ISPA.2012.69
110. Copie A, Fortis TF, Munteanu VI (2013b) Datastores in cloud governance. Int J Comput Commun Control 8(1): 42–49. http://univagora.ro/jour/index.php/ijccc/article/view/167/
111. Copie A, Fortis TF, Munteanu VI (2013a) Data security perspectives in the framework of cloud governance In: EuroPar 2012 Workshops – BDMC 2012, no. 7640 in Lecture Notes in Computer Science. Springer, pp 24–33. http://dx.doi.org/10.1007/978-3-642-36949-0_4
112. Bernstein D, Vij D, Diamond S (2011) An intercloud cloud computing economy - technology, governance, and market blueprints In: Annual SRII Global Conference (SRII 2011), pp 293–299. doi:10.1109/SRII.2011.40, http://dx.doi.org/10.1109/SRII.2011.40
113. Fortiş TF, Munteanu VI, Negru V (2012c) Towards a service friendly cloud ecosystem In: 11th International Symposium on Parallel and Distributed Computing (ISPDC 2012). IEEE Computer Society Press, pp 172–179. http://dx.doi.org/10.1109/ISPDC.2012.31
114. Frîncu ME (2010) Scheduling service oriented workflows inside clouds using an adaptive agent based approach. In: Furht B, Escalante A (eds) Handbook of Cloud Computing. Springer, pp 159–182. doi:10.1007/978-1-4419-6524-0_7, http://dx.doi.org/10.1007/978-1-4419-6524-0_7
115. Frîncu ME, Villegas NM, Petcu D, Muller HA, Rouvoy R (2011) Self-healing distributed scheduling platform In: 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2011). IEEE Computer Society Press, pp 225–234. doi:10.1109/CCGrid.2011.23, http://dx.doi.org/10.1109/CCGrid.2011.23
116. Frîncu ME (2012) Scheduling highly available applications on cloud environments. Future Generation Comput Syst 29. doi:10.1016/j.future.2012.05.017, http://www.sciencedirect.com/science/article/pii/S0167739X12001136
117. Frîncu ME, Crăciun C (2011) Multi-objective meta-heuristics for scheduling applications with high availability requirements and cost constraints in multi-cloud environments In: 4th IEEE International Conference on Utility and Cloud Computing (UCC 2011). IEEE Computer Society Press, pp 267–274. doi:10.1109/UCC.2011.43, http://dx.doi.org/10.1109/UCC.2011.43

118. Di Martino B, Aversa R, Venticinque S, Buonanno L (2011a) Competitive p2p scheduling of users' jobs in cloud In: 2nd Internat. Conference on Cloud Computing, Grids, & Virtualization (Cloud Computing 2011). IARIA, pp 105–112. http://www.thinkmind.org/download.php?articleid=cloud_computing_2011_4_40_20137
119. Rak M, Venticinque S, Máhr T, Echevarria G, Esnal G (2011c) Cloud application monitoring: The mosaic approach In: 3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2011). IEEE Computer Society Press, pp 758–763. http://dx.doi.org/10.1109/CloudCom.2011.117
120. Stankovski V, Petcu D (2013) Developing a model driven approach for engineering applications based on mosaic. Cluster Computing. http://dx.doi.org/10.1007/s10586-013-0263-x
121. Moscato F, Di Martino B, Aversa R (2012b) Enabling model driven engineering of cloud services by using mosaic ontology. Scalable Comput: Pract Exp 13(1): 29–47. http://www.scpe.org/index.php/scpe/article/download/765/345
122. Moscato F, Aversa R, Amato A (2012a) Describing cloud use case in metamorp(h)osy In: 6th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS 2012). IEEE Computer Society Press, pp 793–798. doi:10.1109/CISIS.2012.143, http://dx.doi.org/10.1109/CISIS.2012.143
123. Cloud@Home project. http://cloudathome.unime.it/
124. Amicas project. http://amicas.hpc.uvt.ro
125. Aversa R, Bruneo D, Cuomo A, Martino B, Distefano S, Puliafito A, Rak M, Venticinque S, Villano U (2011a) Cloud@home: Performance management components. In: Guarracino M, Vivien F, Träff J, Cannatoro M, Danelutto M, Hast A, Perla F, Knüpfer A, Di Martino B, Alexander M (eds) Euro-Par 2010 Parallel Processing Workshops, Lecture Notes in Computer Science, vol 6586. Springer, pp 579–586. doi:10.1007/978-3-642-21878-1_71, http://dx.doi.org/10.1007/978-3-642-21878-1_71
126. Distefano S, Puliafito A, Rak M, Venticinque S, Villano U, Cuomo A, Di Modica G, Tomarchio O (2011) Qos management in cloud@home infrastructures In: International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC 2011). IEEE Computer Society Press, pp 190–197. http://dx.doi.org/10.1109/CyberC.2011.40
127. Cuomo A, Di Modica G, Distefano S, Puliafito A, Rak M, Tomarchio O, Venticinque S, Villano U (2013a) An sla-based broker for cloud infrastructures. J Grid Comput 11: 1–25. http://dx.doi.org/10.1007/s10723-012-9241-4
128. Rak M, Cuomo A, Villano U (2011a) Chase: An autonomic service engine for cloud environments In: 20th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2011). IEEE Computer Society Press, pp 116–121. http://dx.doi.org/10.1109/WETICE.2011.21
129. Petcu D (2012e) Towards programmable infrastructures: the steps made by cloud computing and their technical support In: WoSS-4, CLASS Conference. Slovenia, pp 19–21. http://www.kc-class.eu/datoteke/proceedings-woss-4.pdf
130. Petcu D, Frîncu ME, Panica S, Neagul M (2012a) Towards programmatic management of services from multiple clouds In: 4th International Conference on Intelligent Networking and Collaborative Systems (INCoS 2012). IEEE Computer Society Press, pp 487–488. http://dx.doi.org/10.1109/iNCoS.2012.77
131. Calcavecchia NM, Căprărescu BA, Di Nitto E, Dubois DJ, Petcu D (2012) Depas: A decentralized probabilistic algorithm for auto-scaling. Computing 94: 701–730. doi:10.1007/s00607-012-0198-8, http://dx.doi.org/10.1007/s00607-012-0198-8
132. Căprărescu BA, Petcu D (2012) Decentralized probabilistic auto-scaling for heterogeneous systems In: 4th International Conference on Adaptive and Self-Adaptive Systems and Applications (Adaptive 2012), pp 7–12. http://www.thinkmind.org/download.php?articleid=adaptive_2012_1_20_50022
133. Căprărescu BA, Kaslik E, Petcu D (2012) Theoretical analysis and tuning of decentralized probabilistic auto-scaling. CoRR abs/1202.2981, http://arxiv.org/abs/1202.2981
134. MODAClouds project. http://www.modaclouds.eu
135. Ardagna D, di Nitto E, Mohagheghi P, Mosser S, Ballagny C, D'Andria F, Casale G, Matthews P, Nechifor CS, Petcu D, Gericke A, Sheridan C (2012) Modaclouds: A model-driven approach for the design and execution of applications on multiple clouds In: ICSE Workshop on Modeling in Software Engineering (MISE 2012), pp 50–56. doi:10.1109/MISE.2012.6226014, http://dx.doi.org/10.1109/MISE.2012.6226014
136. Helix Nebula project. http://helix-nebula.eu
137. Aversa R, Di Martino B, Rak M, Venticinque S, Villano U (2011c) Performance prediction for hpc on clouds In: Cloud Computing. John Wiley & Sons, Inc, pp 437–456. doi:10.1002/9780470940105.ch17, http://dx.doi.org/10.1002/9780470940105.ch17
138. HOST project. http://host.hpc.uvt.ro
139. Research potential of convergence regions. http://cordis.europa.eu/fp7/capacities/convergence-regions_en.html
140. Drozdowicz M, Wasielewska K, Ganzha M, Paprzycki M, Attaoui N, Lirkov I, Olejnik R, Petcu D, Bădică C (2011) Ontology for contract negotiations in an agent-based grid resource management system. In: Iványi P, Topping BHV (eds) Trends in Parallel, Distributed, Grid and Cloud Computing for Engineering. Saxe-Coburg Publications, pp 335–354. doi:10.4203/csets.27.15
141. Wasielewska K, Ganzha M, Paprzycki M, Drozdowicz M, Petcu D, Badica C, Attaoui N, Lirkov I, Olejnik R (2011) Negotiations in an agent-based grid resource brokering system. In: Iványi P, Topping BHV (eds) Trends in Parallel, Distributed, Grid and Cloud Computing for Engineering. Saxe-Coburg Publications, pp 355–374. doi:10.4203/csets.27.16
142. Šperka S, Smrž P (2012) Towards adaptive and semantic database model for rdf data stores In: 6th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2012). IEEE Computer Society Press, pp 810–815. http://dx.doi.org/10.1109/CISIS.2012.137
143. Olaii. http://www.olaii.com
144. SEED project. http://www.seed-project.eu
145. Competitiveness and Innovation Framework Programme. http://ec.europa.eu/cip/
146. Petcu D (2011) Portability and interoperability between clouds: Challenges and case study. In: Abramowicz W, Llorente IM, Surridge M, Zisman A, Vayssière J (eds) Towards a Service-Based Internet, Lecture Notes in Computer Science, vol 6994. Springer, pp 62–74. doi:10.1007/978-3-642-24755-2_6, http://dx.doi.org/10.1007/978-3-642-24755-2_6
147. Jclouds. http://www.jclouds.org
148. Libcloud. http://libcloud.apache.org
149. OCCI. http://occi-wg.org
150. Deltacloud. http://deltacloud.apache.org
151. Open Virtualization Format. http://www.dmtf.org/standards/ovf
152. Cloud Data Management Interface. http://cdmi.sniacloud.com
153. Cloud Infrastructure Management Interface – Common Information Model. http://dmtf.org/sites/default/files/standards/documents/DSP0264_1.0.0.pdf
154. Unified Cloud Interface Project. http://code.google.com/p/unifiedcloud
155. CloudML. http://www.cloudml.org
156. Hogan M, Liu F, Sokol A, Tong J (2011) Nist cloud computing standards roadmap-version 1.0. Tech. rep. NIST Special Publication 500–291. http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909024
157. Ferrer AJ, Hernández F, Tordsson J, Elmroth E, Ali-Eldin A, Zsigri C, Sirvent R, Guitart J, Badia RM, Djemame K, Ziegler W, Dimitrakos T, Nair SK, Kousiouris G, Konstanteli K, Varvarigou T, Hudzia B, Kipp A, Wesner S, Corrales M, Forgó N, Sharif T, Sheridan C (2012) Optimis: A holistic approach to cloud service provisioning. Future Generation Comput Syst 28(1): 66–77. doi:10.1016/j.future.2011.05.022, http://dx.doi.org/10.1016/j.future.2011.05.022
158. Grozev N, Buyya R (2012) Inter-cloud architectures and application brokering: Taxonomy and survey. Softw Pract Exp. doi:10.1002/spe.2168, http://dx.doi.org/10.1002/spe.2168
159. Simple Cloud API. http://www.simplecloud.org
160. SAGA – a simple api for grid applications. http://saga-project.github.com
161. RightScale. http://www.rightscale.com
162. Enstratius. http://www.enstratius.com/
163. Kaavo. http://www.kaavo.com
164. Aeolus. http://www.aeolusproject.org
165. OPTIMIS Toolkit v2. http://www.optimis-project.eu/Toolkit_v2

166. Petcu D (2012d) A panorama of cloud services. Scalable Computing: Pract Exp 13(4): 303–314
167. SpotCloud. http://spotcloud.com
168. WSO2 Stratos. http://wso2.com/cloud/stratos
169. Petcu D, Stankovski V (2012) Towards cloud-enabled business process management based on patterns, rules and multiple models In: 10th IEEE Internat. Symposium on Parallel & Distributed Processing with Applications (ISPA 2012). IEEE Computer Society, pp 454–459. http://dx.doi.org/10.1109/ISPA.2012.66
170. Heroku. https://www.heroku.com
171. Crăciun CD (2012) Building Blocks of Scalable Applications. Master's thesis, West University of Timişoara, Computer Science Department. https://github.com/downloads/cipriancraciun/masters-thesis/thesis.pdf
172. Petcu D, Rak M (2013) Open-source cloudware support for the portability of applications using cloud infrastructure services. In: Mahmood Z (ed) Cloud Computing: Methods and Practical Approaches, Computer Communications and Networks. Springer, chap 15, pp 323–341. http://dx.doi.org/10.1007/978-1-4471-5107-4_15
173. Contrail project. http://contrail-project.eu
174. OpenShift. https://www.openshift.com
175. Petcu D (2012c) Invitation to a journey in the era of cloud computing In: European Research Activities in Cloud Computing. Cambridge Scholars Publishing, pp 1–20. http://www.c-s-p.org/Flyers/978-1-4438-3507-7-sample.pdf
176. Petcu D, Vasquez-Poletti JL (2012) European Research on Cloud Computing. Cambridge Scholars Publishing, Cambridge
177. Bessani A, Kapitza R, Petcu D, Romano P, Gogouvitis SV, Kyriazis D, Cascella RG (2012) A look to the old-world sky: Eu-funded dependability cloud computing research. SIGOPS Oper Syst Rev 46(2): 43–56. doi:10.1145/2331576.2331584, http://doi.acm.org/10.1145/2331576.2331584
178. 4CaaSt project. http://4caast.morfeo-project.org/
179. Cloud4SOA project. http://www.cloud4soa.eu
180. Cloud-TM pproject. http://www.cloudtm.eu
181. REMICS project. http://www.remics.eu
182. TClouds project. http://www.tclouds-project.eu
183. VISION Cloud project. http://www.visioncloud.eu

doi:10.1186/2192-113X-2-12
Cite this article as: Petcu *et al.*: **Experiences in building a mOSAIC of clouds.** *Journal of Cloud Computing: Advances, Systems and Applications* 2013 **2**:12.