

ServiceWave, 26-28 October 2011, Poznan
Track: Cloud Computing
Session: Cloud Computing 2

Portability and Interoperability between Clouds. Challenges and Case Study

Dana PETCU,

Institute e-Austria Timisoara, and West University of Timisoara, RO

<http://web.info.uvt.ro/~petcu>



Content

- Problem definition and taxonomy
- Approaches
- Case study: mOSAIC solution

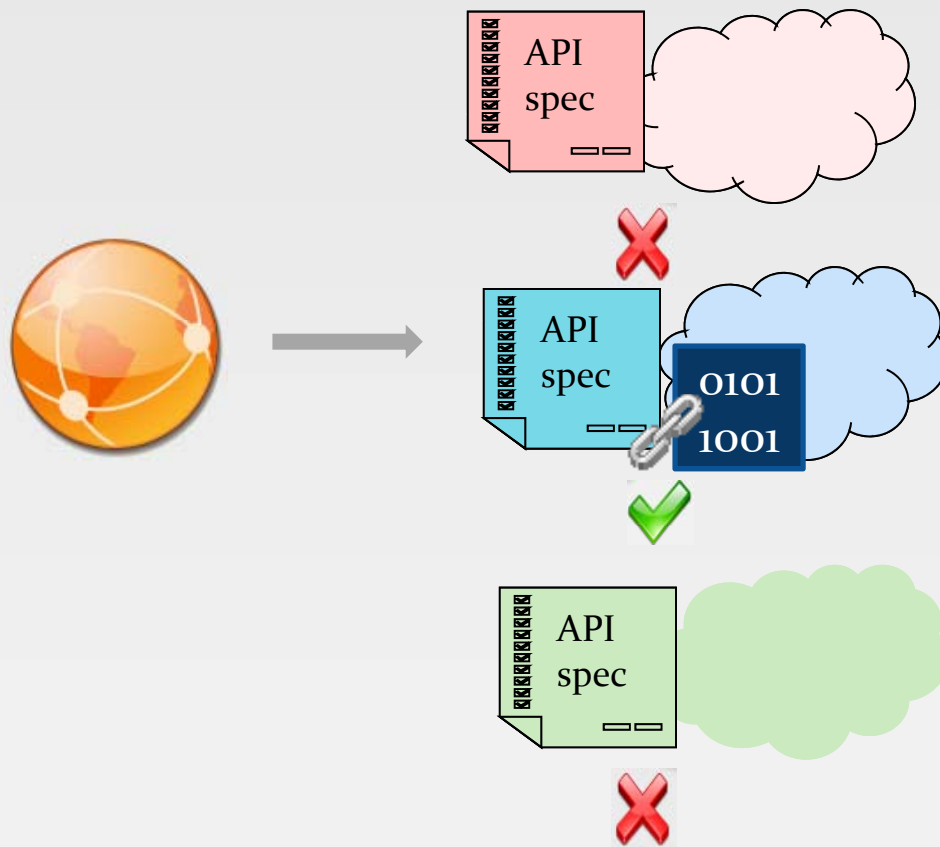


ServiceWave, 26-28 October 2011, Poznan
Track: Cloud Computing
Session: Cloud Computing 2

Problem definition and taxonomy



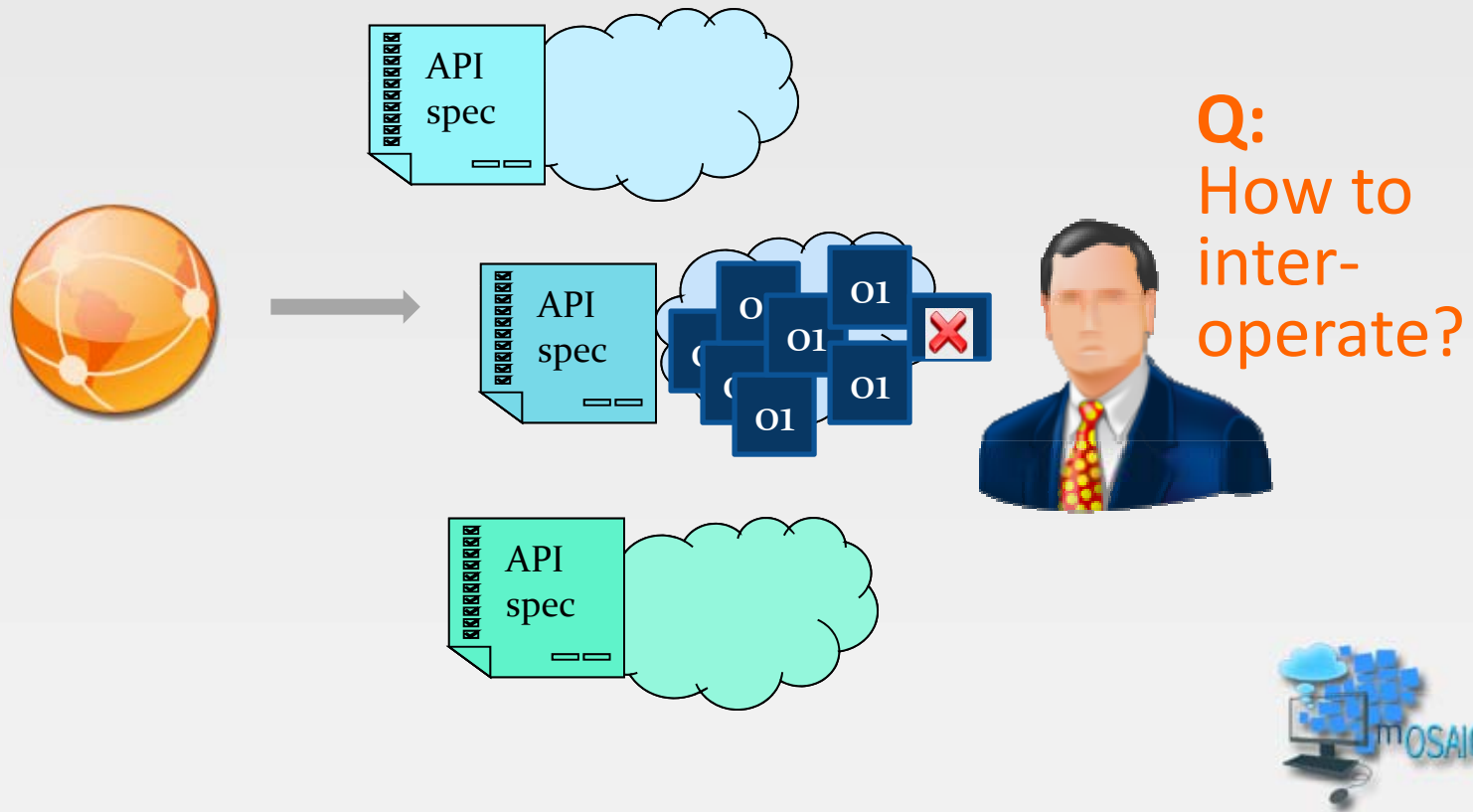
Portability in Clouds?



Q:
How to
port the
appl?



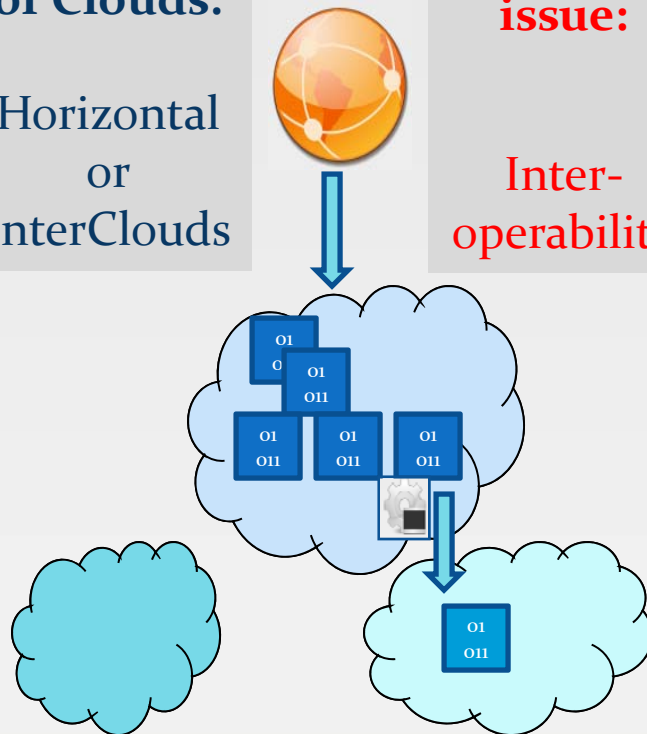
Interoperability in Clouds?



Scenarios for multiple Clouds

Federation
of Clouds:

Horizontal
or
InterClouds



**Main
issue:**

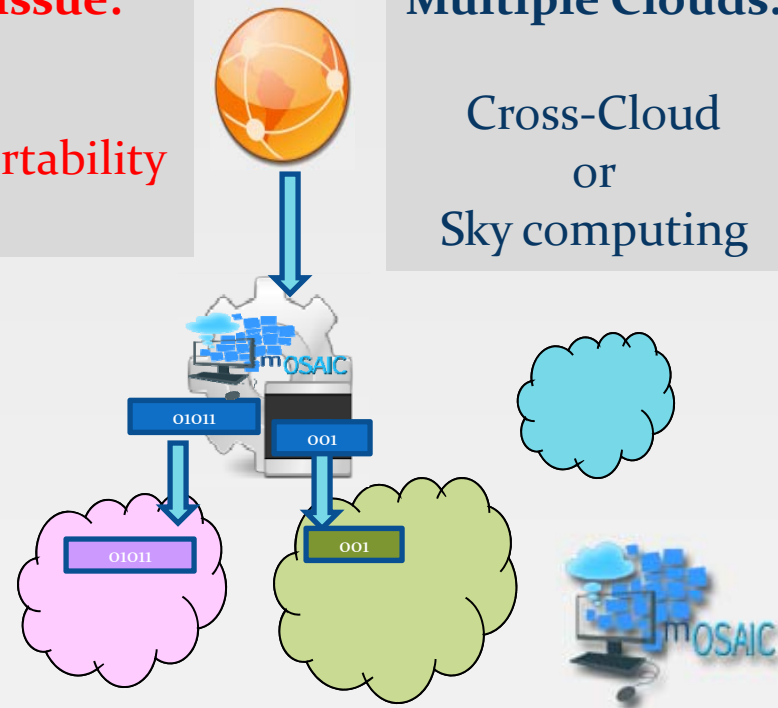
**Inter-
operability**

**Main
issue:**

Portability

On-the-fly
Multiple Clouds:

Cross-Cloud
or
Sky computing



Use cases of multiple Clouds

- **NIST CCSRWG (CC standard, 2011) classification**
 - Serially (one Cloud after another)
 - Migration between Clouds
 - Interface across multiple Clouds
 - Work with a selected Cloud
 - Change Cloud vendors
 - Simultaneously (several Clouds at a time)
- **CC Use Case Discussion Group**
 - Changing Cloud vendor
 - Hybrid Cloud (Distributed deployment?)



Refinement?/distributed deployment

- **Federated Clouds (one trust domain):**
 - Scale out
 - Mutual backup and recovery from a disaster
- **Hybrid Cloud (many trust domains)**
 - Use different cloud services at a time (e.g. tests)
 - Manage selected resources in different clouds
 - Social network applications



Interoperability definition

- **Dictionary:**
 - Property referring to the ability of diverse systems to work together
- **By mottos:**
 - avoid vendor lock-in
 - develop your application once, deploy anywhere
 - enable hybrid clouds
 - one API to rule them all

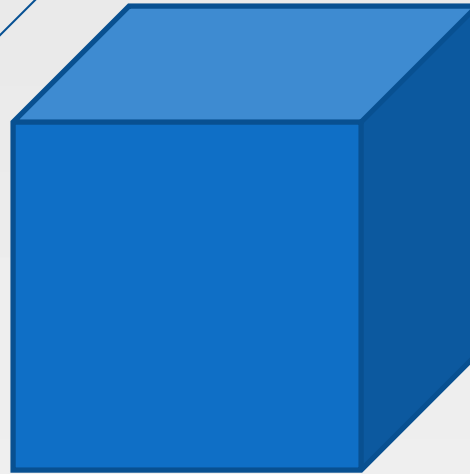


Interoperability/Clouds-dimensions

POLICY:

Federate, communicate
between providers

RUNTIME:
Migration support

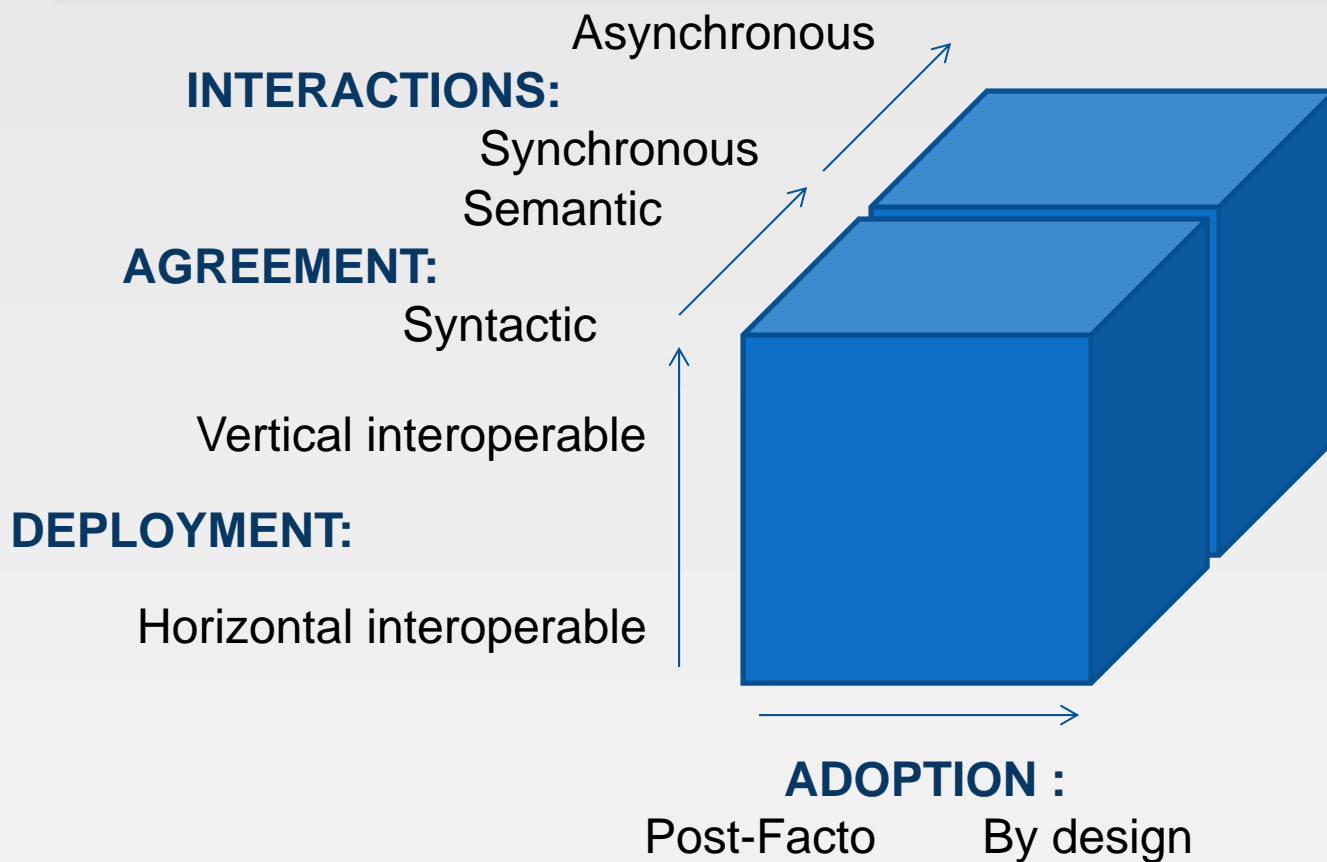


DESIGN:

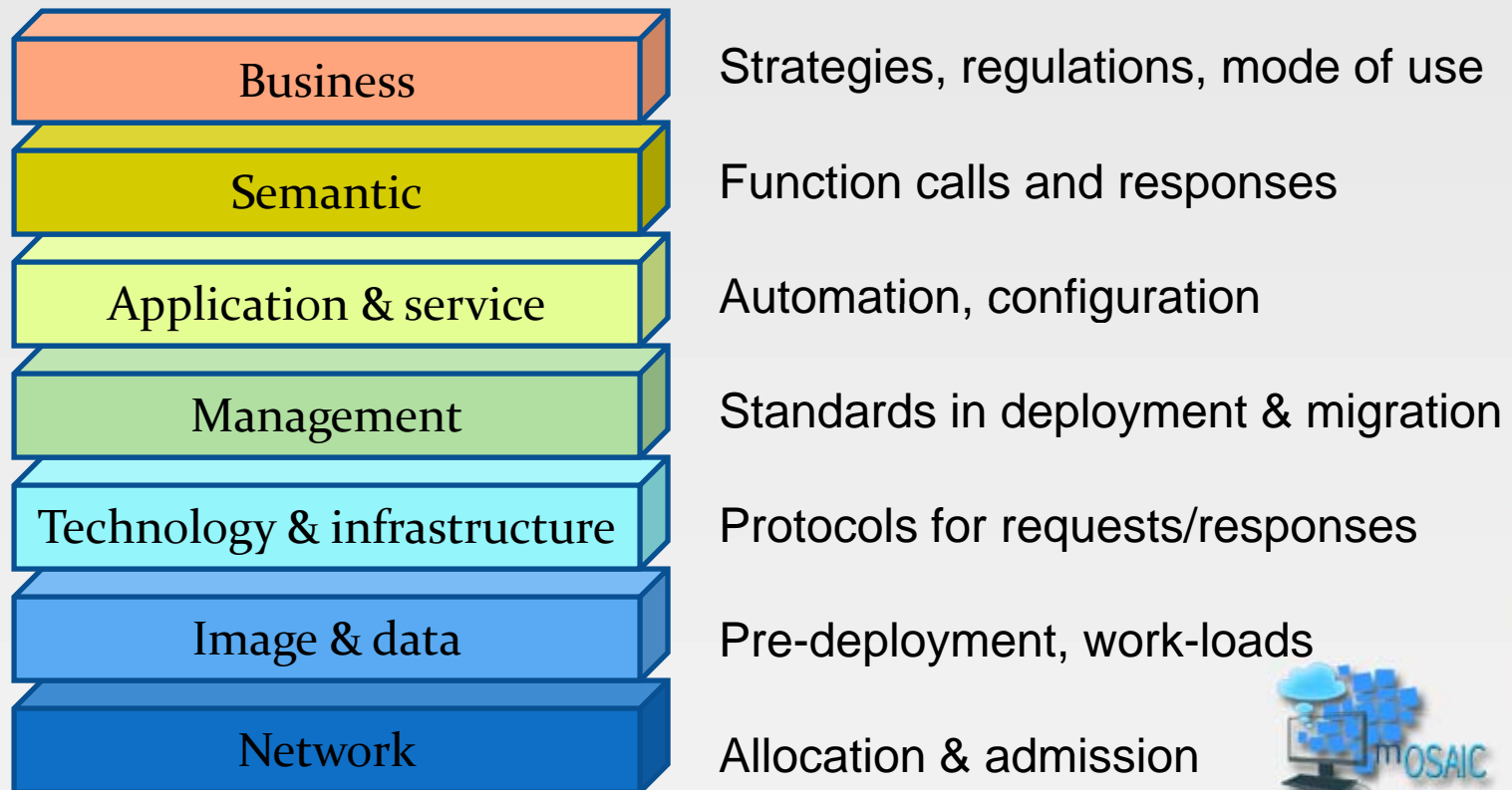
Abstract the programmatic differences



Interoperability/Clouds – types



Interoperability/Clouds-targets



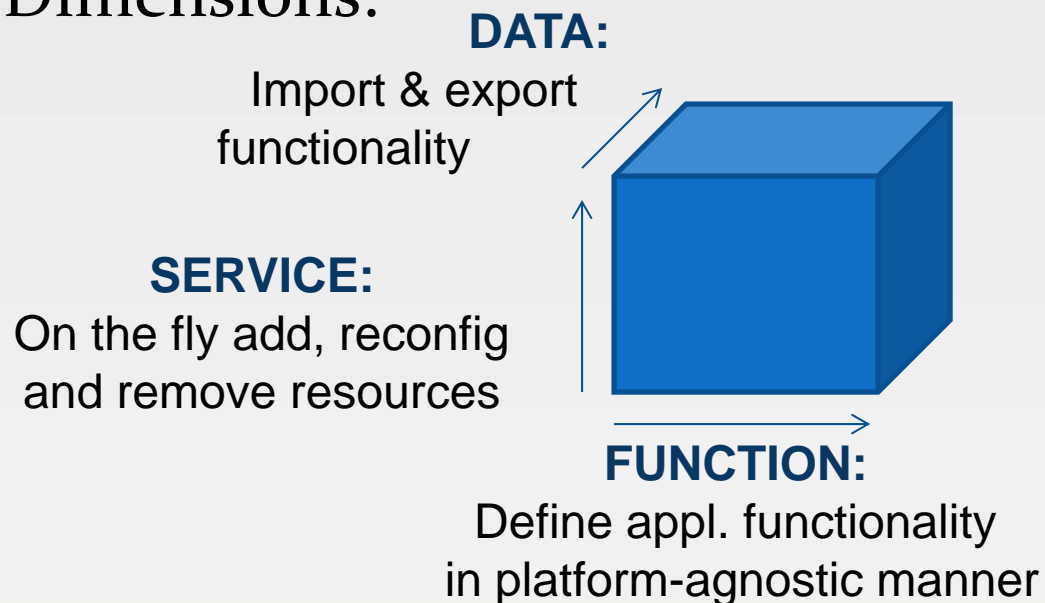
Interoperability/Clouds- history

1. Migration – targets VMs
 - Create, import, share VMs (e.g. use OVF)
2. Federation – targets networking
 - Portable VMs moved between clouds and hypervisors without reconfiguring anything
3. On-demand (burst) – targets APIs
 - Migration and federation on demand
 - Interoperability focused on storage and compute (e.g. CDMI, OCCI)

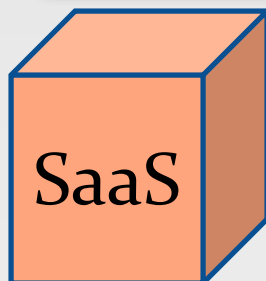


Portability between Clouds

- Ability to use components or systems lying on multiple hardware or software environments
- Dimensions:



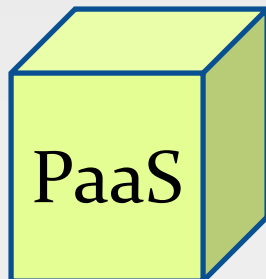
Portability at XaaS level



Preserve/enhance functionality when substitute softw

Measures:

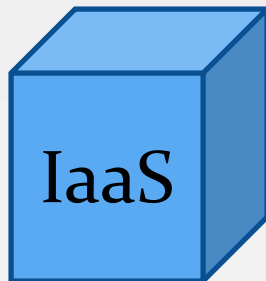
- open source; proprietary/open formats;
- integration techs; appl server/OS



Minim.appl.rewriting while preserve/ enhance control

Measures:

- proprietary vs.open APIs, progr.languages,data formats
- tight vs. loose coupled services
- abstract layers for queuing & messaging



Appls and data migrate and run at a new provider

Measures:

- ability to port VMs and data
- underlying configurations across providers



Requirements

Market	Economic models, cost-effectiveness, license flexibility, negotiated SLAs, leasing mechanisms
Application	Data portability and exchange, scale-out, location-free, workflow management
Programming	Minimal reimplementation when move, standard APIs, same tools for cloud-based and enterprise-based apps
Monitoring	SLA and performance monitoring, QoS aware services, service audit, sets of benchmarks
Deployment	Deploy in multiple clouds with single management tool, navigation between services, automated provisioning, resource discovery and reservation, behavior prediction
AA & Security	Single sign-on, digital identities, security Standards, trust mechanisms, authentication



ServiceWave, 26-28 October 2011, Poznan

Track: Cloud Computing

Session: Cloud Computing 2

Current approaches to Interoperability and Portability between Clouds



Categories

1. Open APIs
2. Open protocols
3. Standards
4. Abstraction layers
5. Semantic repositories
6. Domain specific languages



Open APIs

- **Examples:**

- jClouds, libcloud, Cloud::Infrastructure, SimpleCloud, DaseinCloud, OpenStack, ServerTemplates

- **Counterexample:**

- Microsoft Azure

- **Classification:**

1. API with multiple implementations (e.g. Eucalyptus vs. EC2)
2. API runnable on multiple clouds (MapReduce and Hadoop)
3. Two levels, App-logic layer and Cloud layer (e.g.->)



Open protocols & standards

- **Open protocols**

- Examples: OCCI and Deltacloud
- Counterexamples: vCloud and EC2

- **Standards**

- Examples: OVF/DMTF, CDMI/SNIA
- Groups: CloudAudit, CSA, DMTF, ETSI TC CLOUD, OGF, OMG, OCC, OASIS, SNIA, CCIF, GICTF, ODCA



Abstraction layers

1. Mediators – example:

- use service manifests in Reservoir, service providers as mediators between providers and end-users

2. Frameworks – examples:

- SLA@SOI – services as economic goods
- CSAL – integrated namespace for apps

3. Reference architecture – example:

- RASIC – semantic inter-operability



Semantic & Domain specific languages

- **Semantic:**
 - @levels: interface, component, data
 - Annotate services & appls
 - Example: UCI with implementations for EC2 & ECP
- **Domain specific languages**
 - To create a cloud aware appl or interpret a VM



ServiceWave, 26-28 October 2011, Poznan
Track: Cloud Computing
Session: Cloud Computing 2

Case study: mOSAIC

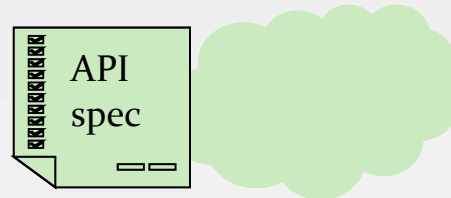
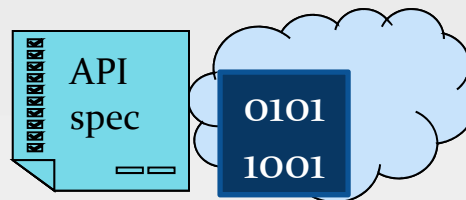
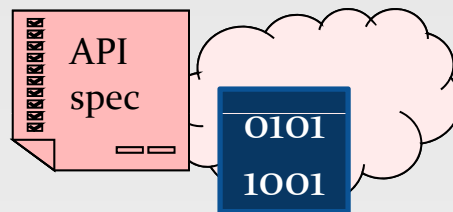
**Open-source
API and Platform
for multiple Clouds**



www.mosaic-cloud.eu



Motto: Fly through Clouds



Answer 1:
Find "A"
Cloud

Answer 2:
Application
portability



mOSAIC promises

- September 2011:** 1st implementation of API
Cloud ontology
- September 2012:** Platform available
- March 2013:** Full software package



Keywords

- Vendor agnostic API
- Open source PaaS
- Cloud broker
- Multi-agent technologies
- SLA negotiations
- Semantic processing
- Multiple Clouds
- Long time running applications
- Component-based applications
- Event driven, asynchronous



API requirements

- Application **portability**
- **Elasticity** at the level of application components
- Freedom to choose the programming **paradigm**
- Build **own stack of software** needed for the application



Vision

- **Based on the Cloud offers:**
 - Hosted systems
 - Deployable Cloud-aware products
- **Novelty:**
 - Build applications on top of deployable Cloud-aware products
 - Support components communications
 - Elasticity to a new level (application dependent)
- **Architectural selection:**
 - Event driven
 - Asynchronous communications



Hosted vs. deployable systems

Table 10: Cloud-related products: Storage and Databases

Category	Type	Instances
Columnar databases	Hosted systems	<ul style="list-style-type: none"> • Amazon SimpleDB; • BigTable;
	Deployable systems	<ul style="list-style-type: none"> • distributed: <ul style="list-style-type: none"> ◦ Cassandra; ◦ Hadoop HBase; ◦ Hypertable; • single-server: <ul style="list-style-type: none"> ◦ M/DB; ◦ MonetDB; ◦ LucidDB;
Key value databases	Hosted systems	<ul style="list-style-type: none"> • Amazon S3; • RackSpace Cloud Files;
	Deployable systems	<ul style="list-style-type: none"> • distributed: <ul style="list-style-type: none"> ◦ Riak; ◦ Membase; ◦ kumofs; ◦ LightCloud; ◦ GridFS; ◦ Voldemort; ◦ Redis; ◦ Hibarj; ◦ Kai; ◦ Ringo; ◦ Dynomite; • single-server: <ul style="list-style-type: none"> ◦ MemcacheDB; ◦ MemcacheD; ◦ Tokyo Cabinet; ◦ Kyoto Cabinet; ◦ BerkeleyDB; ◦ GT.M;
Document databases	Deployable systems	<ul style="list-style-type: none"> • MongoDB; • CouchDB;
Distributed file systems	Deployable systems	<ul style="list-style-type: none"> • Hadoop HDFS; • Tahoe-LAFS; • Ceph; • Sector; • CloudStore; • MooseFS; • mogilefs;
Distributed atomic databases	Deployable systems	<ul style="list-style-type: none"> • Zookeeper; • Scalaris; • Keyspace;
Distributed hash tables	Models	<ul style="list-style-type: none"> • Kademlia • Pastry • Tapestry • Chord



Type of applications

- **API support:**

- **Long-running scalable applications (“new”)**

- Run for undefined period of time
 - Scale upward and downward as the usage demands it
 - Composed by multiple and communicating components
 - Examples: web-crawlers, forecasting systems (financial, meteorology), economical applications (ERP) etc

- **Supplementary platform support:**

- **Massive batch processing (“legacy”)**

- Run for defined period of time
 - Data or computational intensive applications



Basic concepts

- **Cloud Building Block (CBB):**
 - basic component of an application
 - can be CR or CC
- **Cloud Resource (CR):**
 - CBB controlled by Cloud provider
- **Cloud Component (CC):**
 - configurable CBB controlled by application developer
 - CC instances consume CR
 - communication between CC via CR like message queues (control redirection in case faults, scale-up)
 - example: a virtual appliance (e.g. web server + appl server + customized e-commerce appl)



Cloud components (CC) properties

- **Elastic**
 - scale up and down no.of instances of the same CC
- **Manageable**
 - Possible to configure it and change the parameters
- **Isolated**
 - CC instances independent from other CC
- **Fault tolerant**
 - Automated using the Container (instance manager)
- **Implemented by**
 - a **Container** + several **Cloudlets** instances



Clouplet and Connector

- **Clouplet:**

- Behavior: event-driven, stateless
- Elasticity:
 - no. of Clouplet instances controlled by Container
 - no. of Containers
- Functionality do not depend on no. instances
- Its backend interface: defines the set of events to which the Clouplet should react

- **Connector:**

- Behavior: RPC
- Abstract the access to Cloud resources

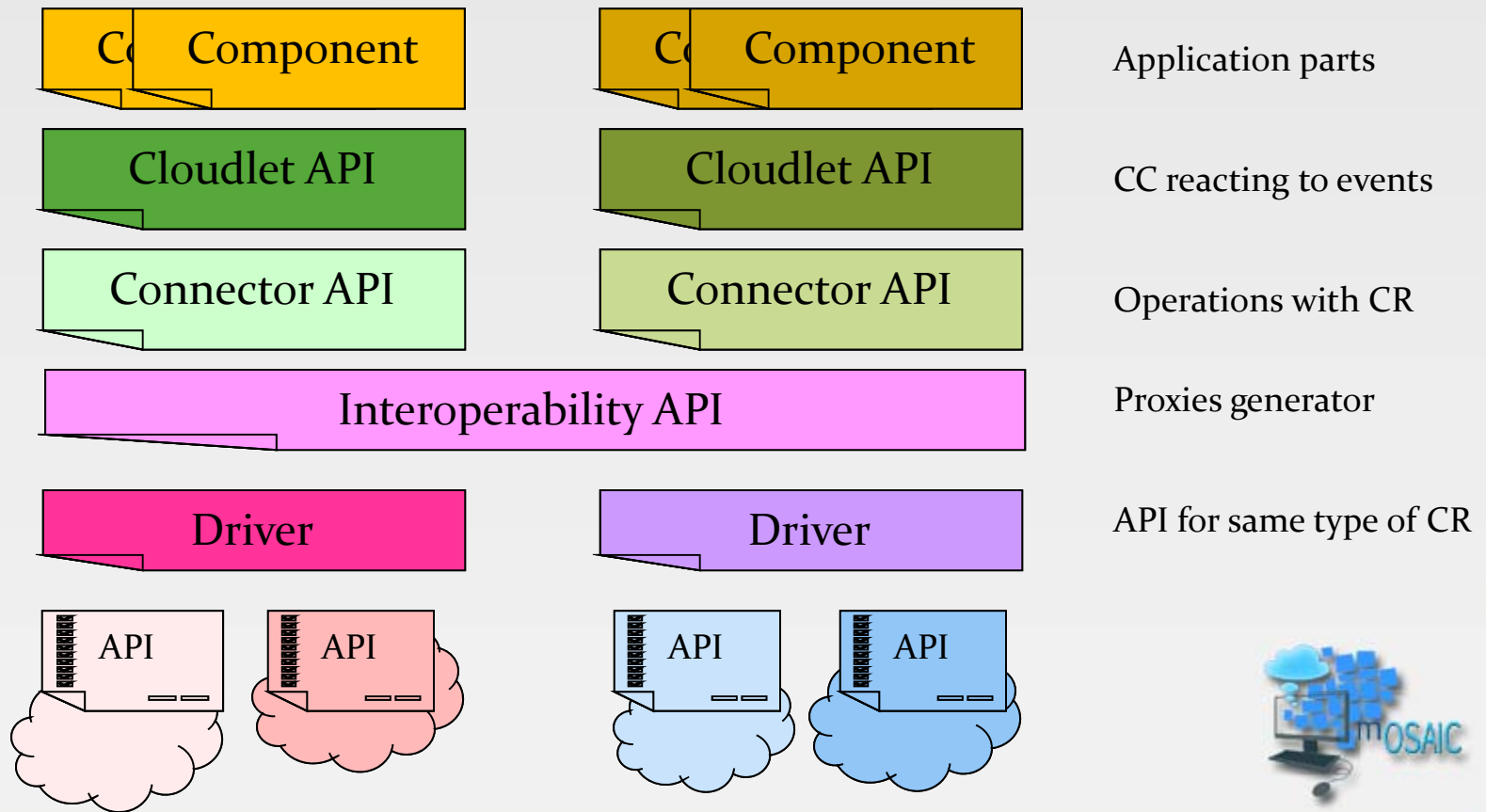


Interoperability API and Drivers

- **Interoperability API**
 - Ensure language independence
 - Protocol syntax and semantic enforcements
 - RPC solution
 - Stubs to Driver API and Proxies to Connector
- **Driver API**
 - Wraps the native API
 - All resources of the same type are exposed with the same interface
 - Eg. a Membase vs. a Riak key-value store: a matter of configuration.



API layers



Details:

- **Selected papers/mOSAIC in 2011:**
 - **API layers:** *Towards a cross-platform Cloud API*, CLOSER 2011, SciTePress.
 - **API interoperability:** *Building an Interoperability API for Sky Computing*, InterCloud/HPCS, IEEE CS
 - **Platform services:** *Architecting a Sky Computing Platform*, ServiceWave Workshops 2010, LNCS 6569
 - **Test apps:** *Building a Mosaic of Clouds*, EuroPar Workshops 2010, LNCS 6586
- **API user guide:**
 - <https://bitbucket.org/mosaic/mosaic-api/src/4e52732943cc/mosaic-mvn/doc/mOSAICProgrammingGuide.pdf>
- **Current platform - demo:**
 - http://web.info.uvt.ro/~petcu/demo_mosaic_19102011.avi



What's next?

- Techs to sustain the on-the-fly usage of multiple Clouds
 - Multiple brokers and e-Markets
 - Providers that are negotiating
 - Economics models
 - Open PaaS for multiple Clouds
- Agree to:
 - federate at least for solving public servants problems
 - build and adopt standards, open APIs and protocols
 - find a path to help transforming legacy appls



mOSAIC partners



- Second University of Naples, Italy
- Institute e-Austria Timisoara, Romania
- European Space Agency, France
- Terradue SRL, Italy
- AITIA International Informatics, Hungary
- Tecnalia, Spain

From 1st Sept 2011:

- Xlab, Slovenia
- University of Ljubljana, Slovenia
- Brno University of Technology, Czech Republic



www.mosaic-cloud.eu

