

Dana Petcu, Ciprian Craciun, Marian Neagul, Silviu Panica

Institute e-Austria Timisoara, Romania

**Beniamino di Martino, Salvatore Venticinque,
Massimiliano Rak, Rocco Aversa**

Second University of Naples, Italy



ARCHITECTURING A SKY COMPUTING PLATFORM

CONTENT

- SOTA and motivation
- Concepts
 - mOSAIC promises
 - Architectural details
- Implementation plan
 - API
 - Deployment
- Conclusions



HOW WE DEVELOP A CLOUD-BASED APPLICATION?

- “Low level” – usage of IaaS
 - e.g. Amazon EC2, Eucalyptus, Sun Cloud, ElasticHosts, FlexiScale, GoGrid, Enomaly, OpenNebula, SliceHost, Nimbus, AppNexus, F5, Tashi, CohesiveFT, Mosso, Joyent, etc
 - APIs offered by IaaS Cloud service providers to create and manage cloud resources, including compute, storage, and networking components
 - Need knowledge about setting an e-Infrastructure
- “High level” – usage of PaaS
 - E.g. Google App Engine, Microsoft Azure Service Platform [or wait for Orleans], Manjrasoft Aneka, Amazon Web Services etc
 - APIs offered for “programmable infrastructure”
 - No need of knowledge about setting an e-Infrastructure
 - Focus on application development



TOWARDS THE USAGE OF MULTIPLE CLOUDS

○ Portability

- At IaaS level? Ongoing task, e.g. OCCI
- At PaaS level? *NO!*

○ Approaches:

- At IaaS level:
 - Migration of VMs between Cloud providers (e.g. Reservoir)
 - Agreements between Cloud providers
 - Communications between Clouds
 - Use same [OCCI] API e.g. for VM deployment and management
- At PaaS level:
 - *Use services from different Clouds?*



SKY COMPUTING

- Emerging paradigm dealing with dynamically provisioning of resources from distributed domains representing several Cloud computing environments

[Keahey Ket al. Sky Computing. IEEE Internet Computing Sept. 2009]

- Key words: dynamic, several Clouds
- Analogy:
 - Cluster computing with Cloud computing
 - Grid computing with Sky computing
- Issues:
 - Develop Cloud application independent from the IaaS level
 - Ensure best offer selection and (re)negotiation
- Implementations
 - Theoretical studies about the offer selections and billing system

12/13/2010

ServiceWave 2010 - OCS workshop



[SOME] REQUIREMENTS / MOSAIC PROPOSALS

- Application portability from one Cloud to another
- Common language between Cloud providers and application developers
- Adoption of a technology depends on its simplicity and similarity with current models
- Mechanism for selection of Cloud providers
- Application-driven resource management
- API at high level independent from the provider
- Open/standard interfaces
- Cloud ontology
- Cloud usage patterns
- Agent technologies
- Based on profiling and monitoring; SLA changes
- With implementation in high level languages

12/13/2010

ServiceWave 2010 - OCS workshop

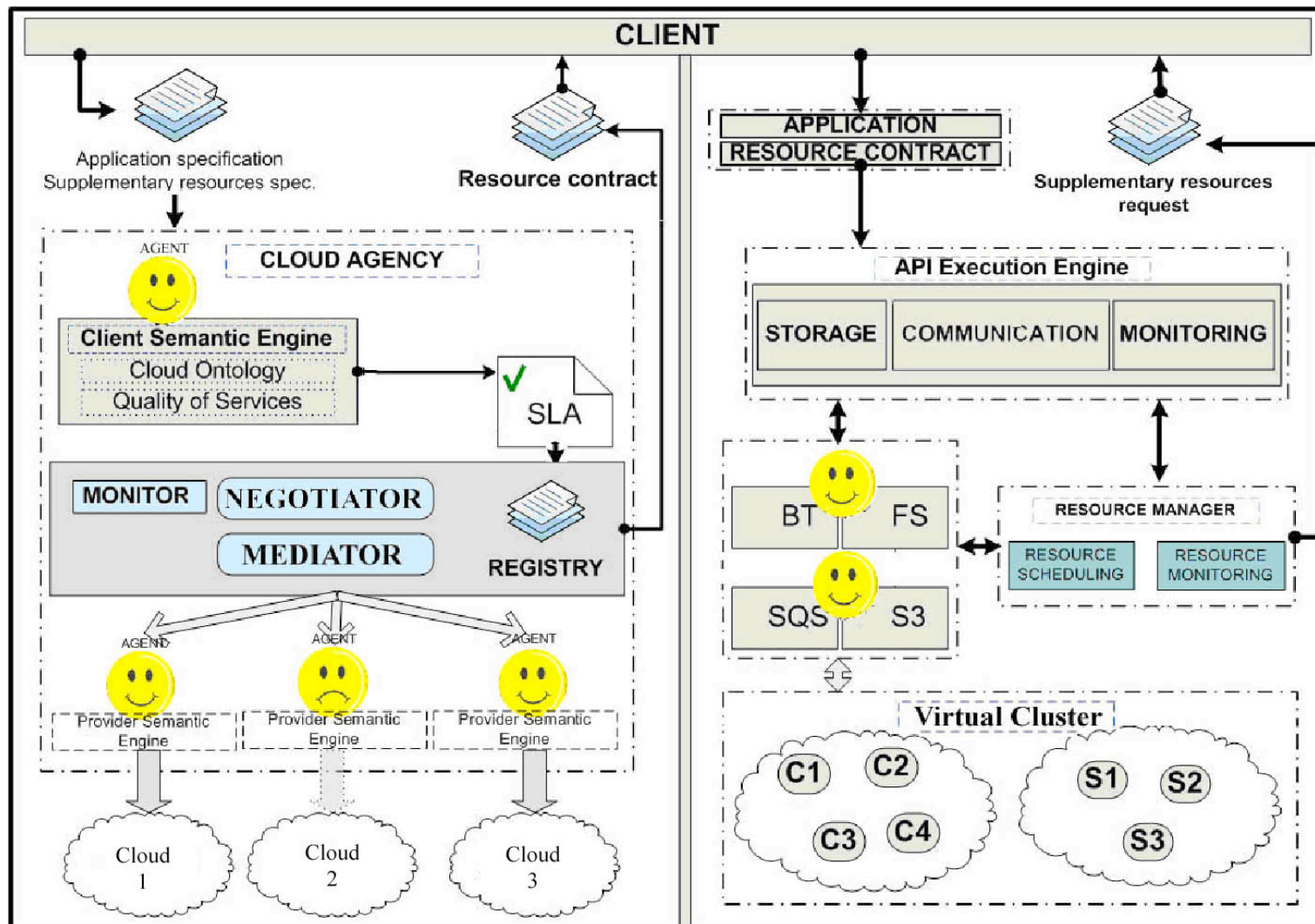


MOSAIC COMMITMENTS

- Title: Open-Source API and Platform for Multiple Clouds
 - **An API**
 - Cloud-based language- and platform-independent API
 - Extends the existing language- or platform-dependent API capabilities with composite features based on *patterns*
 - **A framework**
 - Semantic engine
 - Cloud ontology & Semantic representation of Cloud resources
 - Applications's needs in terms of SLAs and QoS requirements
 - Cloud agency
 - **An open-source platform**
 - a proof-of-the-concept prototype ready to be tested, exploited or extended by its users
 - include instances of the APIs for several programming languages and application tools
- ! Proofs of validity through the use cases and applications



PLATFORM COMPONENTS



TIME LINE

- mOSAIC started at 1st September 2010
- First tasks:
 - Architecture details
 - Cloud usage patterns
 - Own Cloud solution

Month	Achievement
February 2011	Architecture and Cloud usage patterns
August 2011	1 st API implementation, platform usage cases, ontologies
February 2012	2 nd API implementation and framework
August 2012	Platform available, first application package
February 2013	Full software package and proof-of-the-concept applications



STARTING FROM CLOUD USE CASES

- Existing use cases
 - OCCI use cases with IaaS API requirements
 - Cloud Computing Use Case Discussion Group
 - Provider's use cases
 - Research use cases
- mOSAIC's use cases

Type	Title
Data intensive	Storage and data distribution in Earth Observation
	Earth Observation mission reprocessing
	Routine production of Earth Observation products
	Fast data access for crisis situations
	Distributed intelligent maintenance
Compute	Cloud-distributed parameter sweep



CLOUD APPLICATION TYPE

- Main: Long-running scalable applications
 - Run for undefined period of time
 - Scale upward and downward as the usage demands it
 - Composed by multiple components that communicate between them
 - Examples: web-crawlers, forecasting systems (financial, meteorology), economical applications (ERP) etc
- Secondary: Massive batch processing
 - Run for defined period of time
 - Data or computational intensive applications

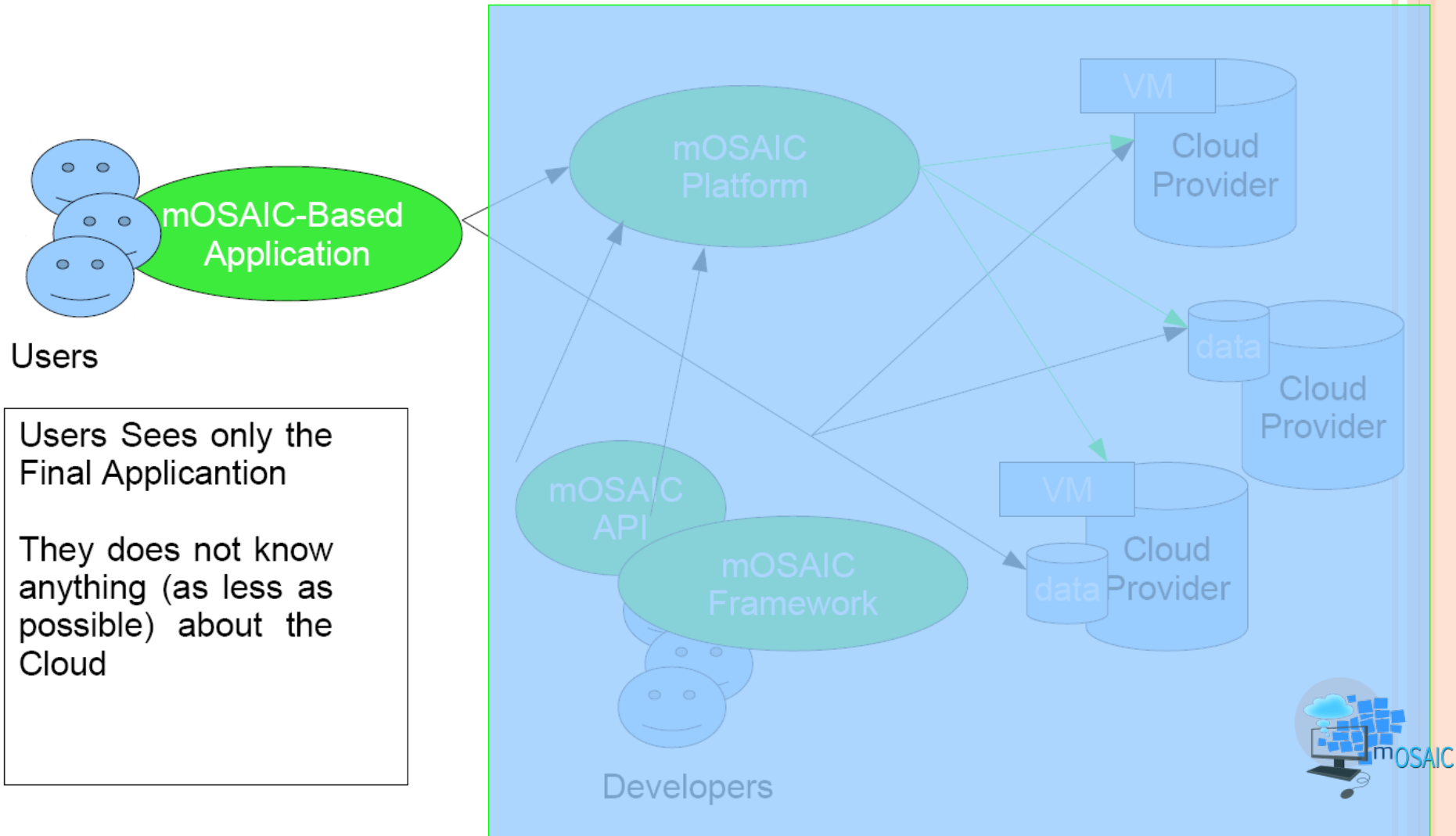


THREE LEVELS OF PROGRAMMABILITY

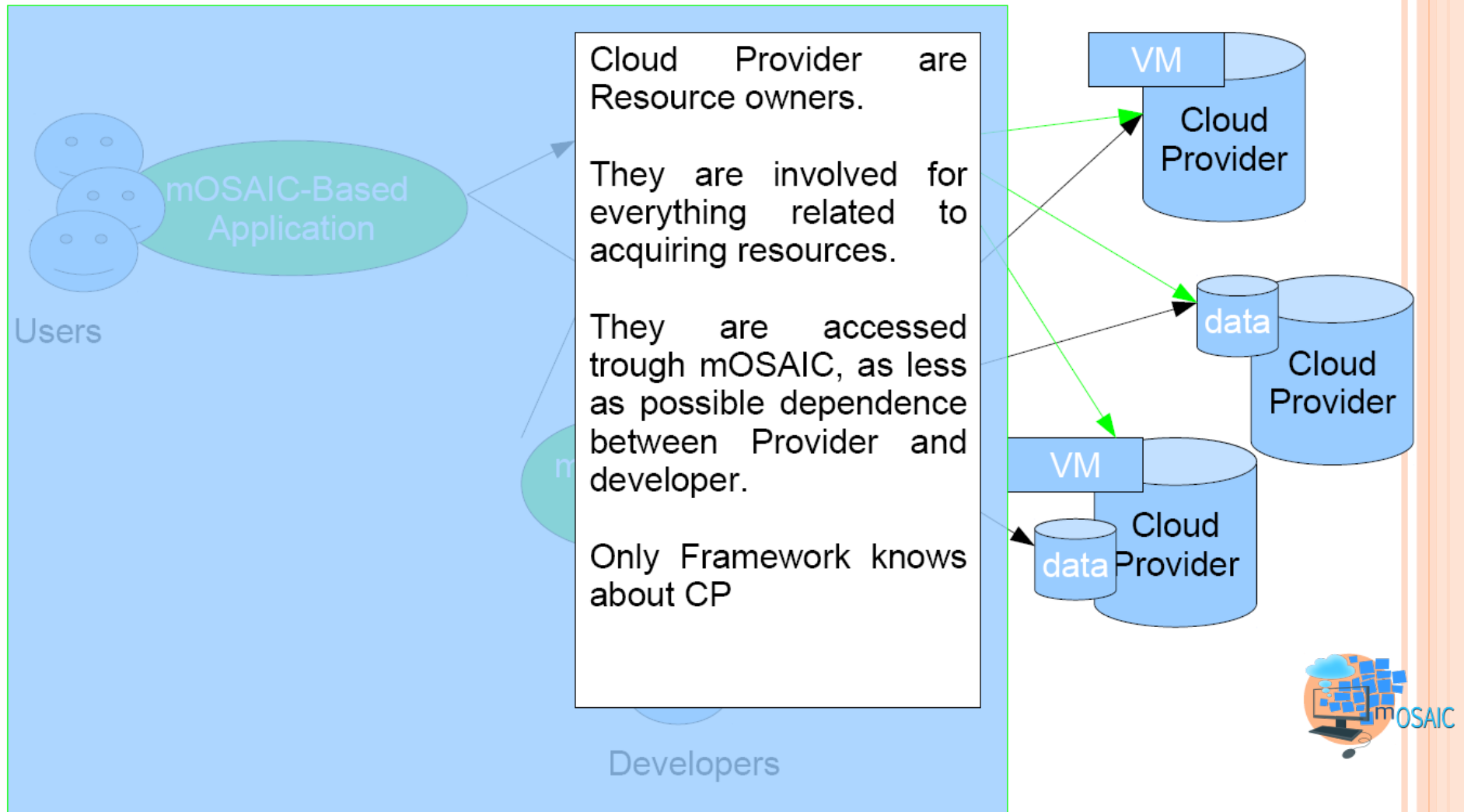
- Low level: Infrastructure Programming Interface
 - For Cloud resources – e.g. VMs, disks, networks, etc
 - Emerging standards to follow: OVF and CDMI
- Middle level: Middleware Programming Interface
 - For interoperability between multiple Clouds
 - Emerging standards to follow: OCCI and UCI
- High level: Application Programming Interface
 - For Cloud services – e.g. data-stores, queues etc
 - Minimal functionalities to be exposed:
 - Storage (distributed FS, block devices, distributed databases)
 - Communication (message queues, RPC, datagram, synchronization)
 - Monitoring
 - Provisioning



USER VIEW @ HIGH LEVEL



PROVIDER VIEW @ LOW LEVEL



PLATFORM VIEW @ MIDDLE LEVEL

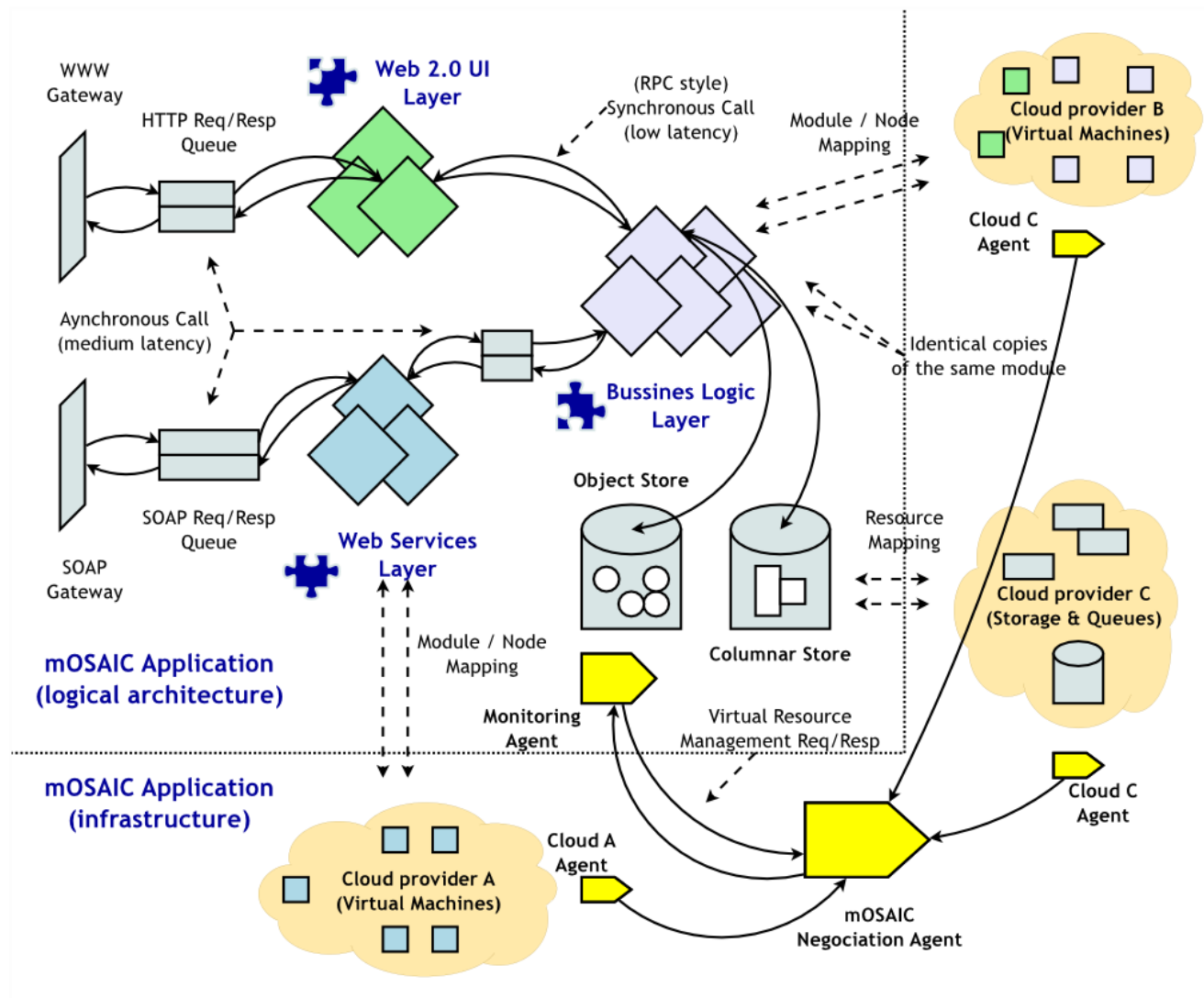
- Decoupling the Development from Deployment !
 - Follow the PaaS concept
 - Developer:
 - Should focus on the development of the application components according to API
 - Platform:
 - Should automate and assist the deployment and execution of the application

=> Views of the same applications:

- Logical one: developers view
- Infrastructure one: platform view



LOGICAL (HIGH) VS. INFRASTRUCTURE (MIDDLE) VIEW



RESTRICTIONS

Steps:

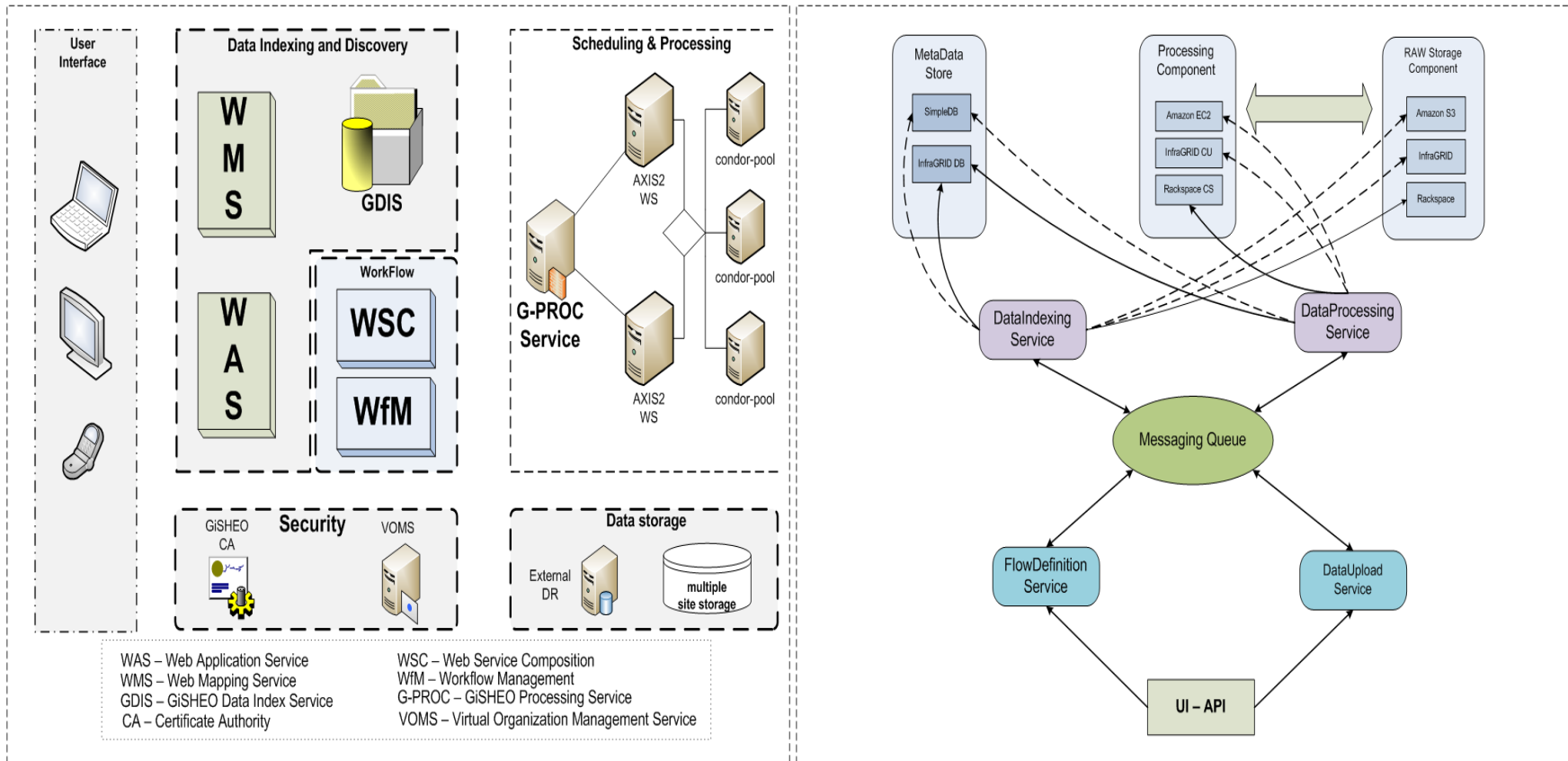
1. Develop components
2. Specify resources reqs
3. Submit reqs to resource broker returning the contract
4. Bootstrap the resources
5. Deploy and start appl
6. Monitor the appl

Guide-lines:

1. Split in components
2. Establish dependencies
3. Use services style
4. Use specific communication patterns
 - RPC, message queues
 - All exchanges (including exterior) through API
 - Avoid sockets

EXAMPLE: ADAPTING A WEB SERVICE PLATFORM

12/13/2010 ServiceWave 2010 - OCS workshop



GiSHEO current architecture
(<http://gisheo.info.uvt.ro>)

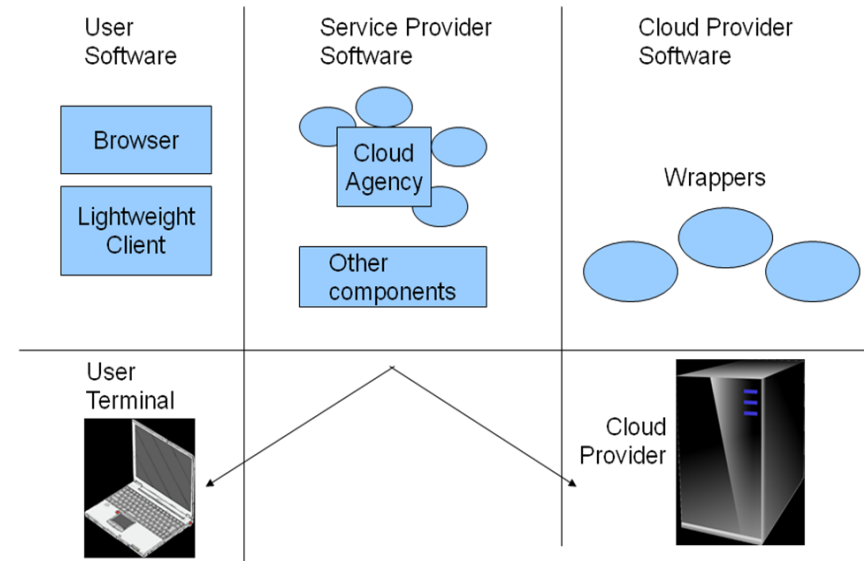
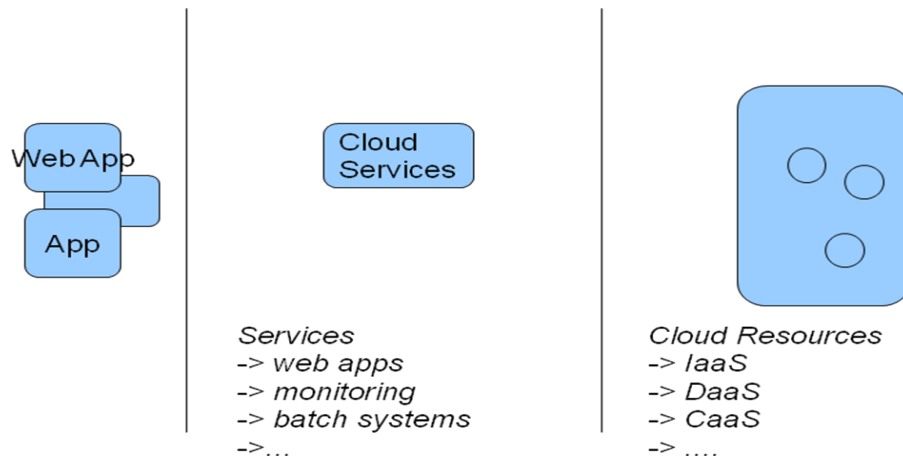
Logic architecture using mOSAIC API



API COMPONENTS

Components application:

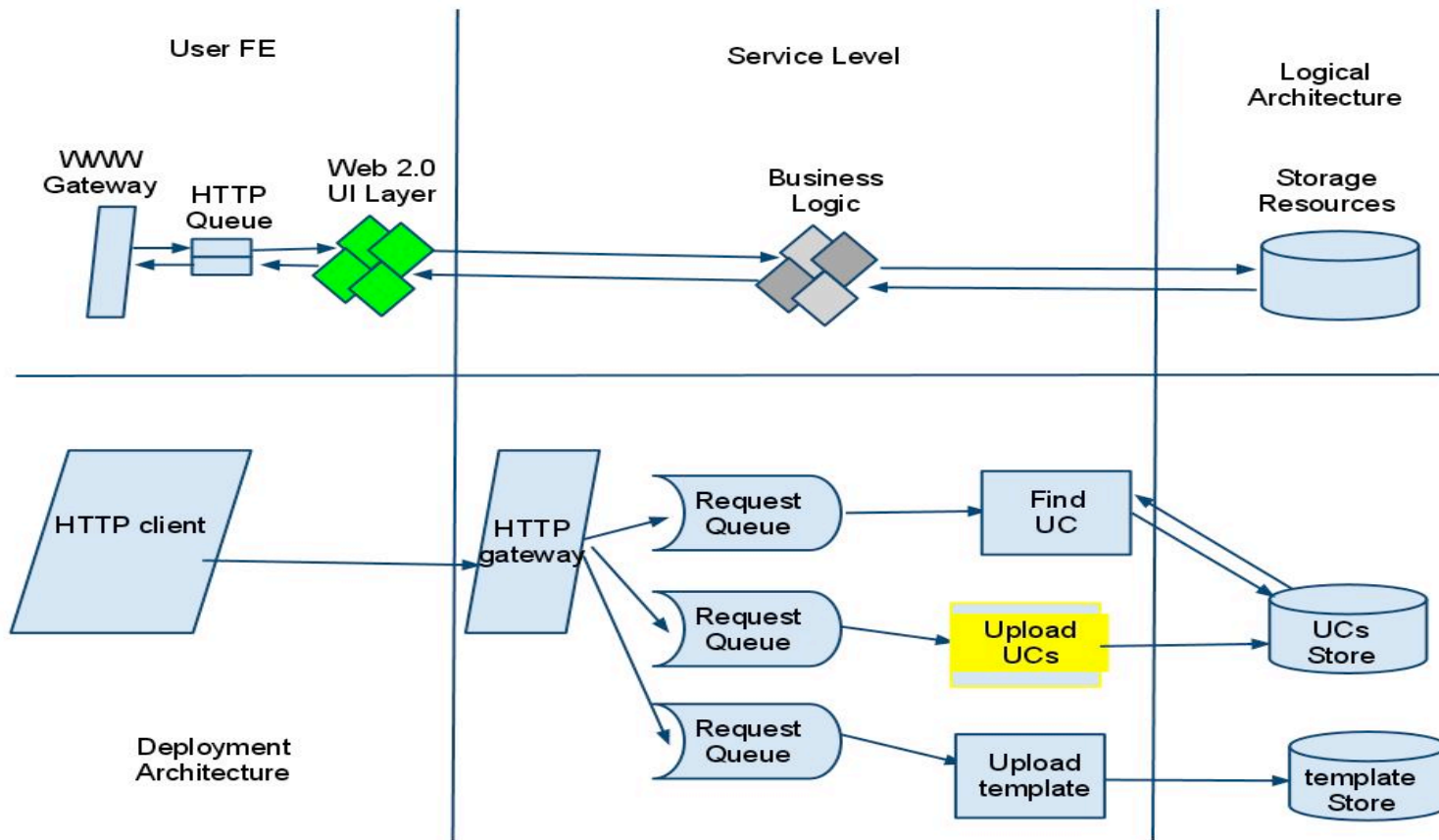
1. View/interface
2. Service behavior
3. Resource management



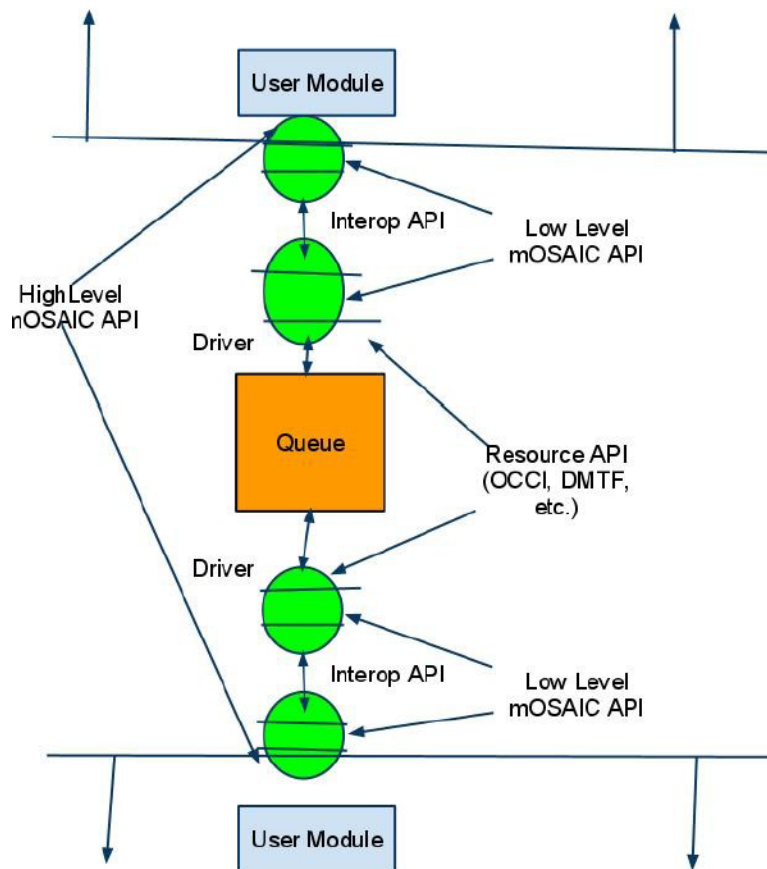
API components:

1. User modules
2. Service modules
3. Provider modules

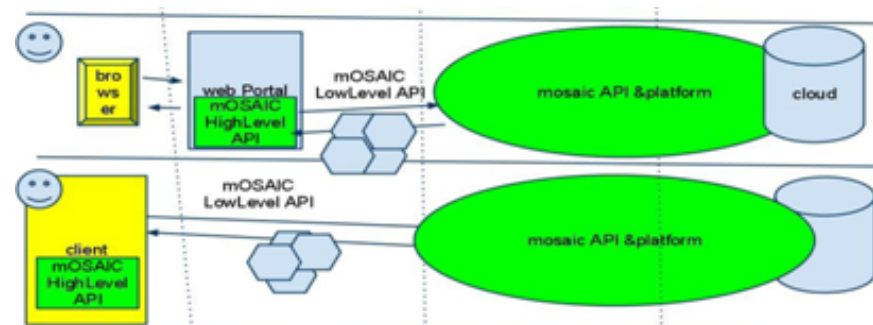
EXAMPLE: UPLOAD/DOWNLOAD WEB APPL



LOW LEVEL API VS. HIGH LEVEL API



Low level: communication
 High level: user module's API



DESCRIPTOR

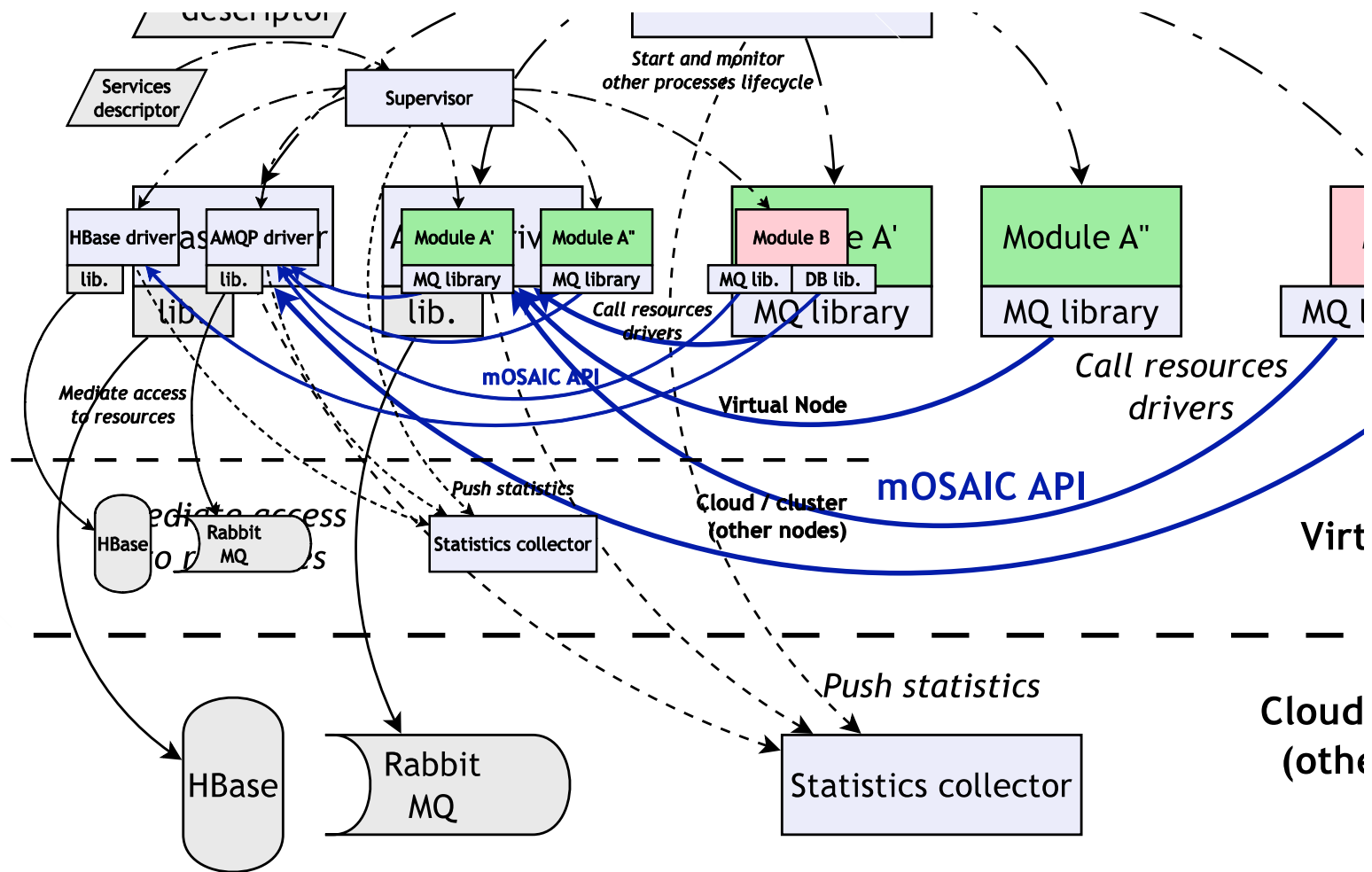
```
<deployment>
  <modules>
    <module type="builtin">
      <identifier> HTTP GW</identifier>
      <filter>
        ...
      </filter>
    </module>
  ...
  <resources>
    <queue>
      <identifier>find-uc-http-requests</identifier>
    </queue>
  ...

```

- The deployment file is derived from the connectivity diagram
- Each component is a module
- Resources (like queue and storage offered by Cloud provider) are declared separately
- Infrastructure information can be added (like the kind of connection between components)



PLATFORM: DEPLOYMENT ON CLOUD RESOURCES



12/13/2010 ServiceWave 2010 - OCS workshop



RELATED WORK

- IaaS integrators for multiple Cloud offers:
 - Reservoir, OpenNebula, Eucalyptus, DeltaCloud, OCCl, Nimbus, libcloud etc
- PaaS integrators for multiple Cloud offers:
 - SimpleCloud targets only PHP apps and 3 providers
 - CloudBroker – negotiation, configuration done manually
 - mOSAIC – not only Web apps, intend larger no. providers, adapt at run time



CONCLUSIONS

- mOSAIC is building open- source solution
 - to support multiple Cloud providers
 - towards application developers
- The proposed solution will offer the developer the freedom of choice in terms of
 - Cloud resource providers and
 - programming environment.



CONTACTS & MATERIALS

- mOSAIC Web site: www.mosaic-cloud.eu
- Partners:
 - Second University of Napoli, Italy
 - Institute e-Austria Timisoara, Romania
 - European Space Agency, France
 - Terradue S.A., Italy
 - Fatronik-Tecnalia, Spain
 - AITIA, Hungary
- Other materials:
 - Cloud Agency & SLA – EuroPar 2010, Workshops: OCI
 - Vision – EuroPar 2010, Workshops: OCI
 - Use cases – Cluster 2010, Workshop: HeteroPar
 - mOSAIC tutorial – SPRERS training event

