**Cloud Systems and Interoperability**

The current cloud-computing view includes many cloud providers, each with its own model, set of services and API (Application Programming Interface), leaving users with an interoperability problem when trying to avoid a potential dependency on a specific provider.

The goal is to deliver distributed systems with the ability of meaningfully exchanging information.

Current approaches tend to tackle this problem (user to cloud or cloud to cloud) by abstracting it, either by providing a common API, which has to map onto each cloud's API, or by introducing brokers that adapt the views of the user and of the cloud [Reference].

The interoperability in the context of multiple clouds has been tackled by active research [8, 37, 38, 39]. The organization of multiple clouds as an intercloud, considered by the IEEE Cloud Computing Initiative, has been described in the literature [40, 41, 42].

Clouds are becoming the new datacenters, now virtualized and much more dynamic. In principle, this means that it should be much easier to change the provider of resources and to discover and use the most convenient services.

Distributed applications, whether supported by a cloud or by a conventional data center, have the same basic interoperability problems.

Existing interoperability frameworks tend to build on existing integration technologies, such as XML, Web Services and RESTful APIs, focusing on how to integrate complex systems by using them.

**What is Interoperabily**

Interoperability, as a means to achieve integration, is as old as networking. Whenever two or more resources need to interact, an interoperability problem arises.

There is no universally accepted definition of interoperability, since its meaning can vary accordingly to the perspective, context and domain under consideration. Although limited to information, the 24765 standard (ISO/IEC/IEEE, 2010) provides the most cited definition of interoperability, as “the ability of two or more systems or components to exchange information and to use the information that has been exchanged”. In other words, merely exchanging information is not enough. Resources must also be able to understand it and to react to it according to each other’s expectations.

The Cloud Computing Use Case Discussion Group defines cloud interoperability in terms of writing code that is able to work with more than one cloud provider simultaneously [46], whereas in [25] it is defined as the ability for multiple cloud providers to work together.

**Partial Interoperability with Compliance and Conformance**

In general, resources are made interoperable by design; they are designed and implemented to work together, allowing the service of one to invoke the service of another. This can be seen in program development and distributed integration, using for example Web Services. In this case:

* Sharing schemas between interacting services and establishing coupling for all the possible values satisfying each schema, even if they are not actually used.
* REST uses data types which usually called media types to be standardized or previously agreed, when they are application specific.
* Searching for an interoperable service is done by schema matching and ontology matching. This does not ensure interoperability and manual adaptations are usually unavoidable.

This section considers the basic interaction between two services and how interoperability can be established. we define interoperability with different perspective, stronger than similarity but weaker than commonality (resulting from using the same schemas and ontologies). The trick is to allow partial (instead of full) interoperability, by considering only the intersection between what the consumer needs and what the provider offers.

Figure Full and Partial

Interoperability (of a consumer with a provider) entails the following properties:

* Compliance (Kokash & Arbab, 2009). The consumer must satisfy (comply with) the requirements established by the provider to accept requests sent to it, without which these cannot be honored.
* Conformance (Kim & Shen, 2007; Adriansyah, van Dongen, & van der Aalst, 2010). The provider must fulfill the expectations of the consumer regarding the effects of a request (including eventual responses), therefore being able to take the form of (to conform to) whatever the consumer expects it to be.

In full interoperability, the consumer can use all of the provider’s capabilities. In partial compatibility, the consumer uses only a subset of those capabilities, which means that compliance and conformance need only be ensured for that subset.

These properties are not commutative for example if A complies with B, B does not necessarily comply with A, but are transitive, so if A complies with B and B complies with C, then A complies with C. Figure 4 illustrates this model. A resource A, in the role of consumer, has been designed for full interoperability with resource B, in the role of provider. A uses only the capabilities that B offers and B offers only the capabilities that A uses. Let us assume that we want to change the provider of A to resource X, which has been designed for full interoperability with resource Y, in the role of consumer. The problem is that A was designed to interact with provider B and X was designed to expect consumer Y. This means that, if we use resource X as a provider of A, B is how A views provider X and Y is how X views consumer A.

There are two necessary conditions to ensure that A is interoperable with X:

Compliance: B must comply with Y. Since A complies with B and Y complies with X, this means that A complies with X, and therefore, A can use X as if it were B, as it was designed for.

* Conformance: Y must conform to B. Since X conforms to Y and B conforms to A, this means that X conforms to A and, therefore, X can replace (take the form of) B without A noticing it.

Partial interoperability has been achieved by subsumption, with the set of capabilities that A uses as a subset of the set of capabilities offered by X. This inclusion relationship, without changing characteristics, is similar in nature to polymorphism, used in many programming languages, but here it applied to a  distributed context. It constitutes the basis for transitivity in compliance and conformance, as well as the mechanism to reduce coupling between two resources to the minimum required by the application.

These properties are not commutative, since the roles of consumer and provider are different and asymmetric by nature. However, nothing prevents two interacting resources from switching roles in a symmetric way, by using and offering capabilities in a reciprocal fashion, which is typical of certain interaction protocols.