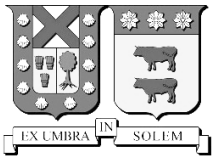
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

TOWARDS A SECURITY REFERENCE ARCHITECTURE FOR FEDERATED INTER-CLOUD SYSTEMS

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF:

MASTER OF SCIENCE (MSc.)
IN INFORMATIC ENGINEERING

Oscar Encina Calquín
December 2014

Department of Informatics

Author:	Oscar Encina Calquín
Title:	Towards a Security Reference Architecture for Inter-Cloud Systems
Institution:	Universidad Técnica Federico Santa María
Advisors:	Dr. Raúl Monge A.
	Dr. Eduardo B. Fernandez
	Dr. Sergio F. Ochoa
	Dr. Hernán Astudillo
Degree:	Master of Science in Informatic Engineering
Year:	2014

ABSTRACT

Inter-cloud systems are the evolution of traditional Cloud Computing systems; allowing the cooperation and aggregation of several services coming from multiple heterogeneous *Service Providers*. A special case of an Inter-cloud system has as purpose the dynamic sharing of resources between *Service Providers* giving just-in-time, opportunistic, and scalable provisioning of services, consistently achieving *QoS* targets under variable conditions. The cooperation between them is achieved by forming a federation between *Service Providers*; this special kind of *Inter-cloud* is called *Federated Inter-cloud*.

Currently there is much divergence related to the *Inter-Cloud* environment. Several components, projects and architectures have been proposed, some of them referring to the same solutions, but with different names, confusing and delaying the advance of these technologies and further advances related to this area. These problems affect also the securing process of the system.

This thesis proposes the creation of a *Reference Architecture* for federated *Inter-cloud* systems by using architectural patterns and also, use this *Reference Architecture* as the basis for the securing process of these kinds of systems.

A *Reference Architecture* is a standardized, generic software architecture in a particular domain and with no platform dependencies. It should define the fundamental components of a system and the interaction among these units, being a useful tool for building and understanding complex systems such as Inter-Clouds. This would catalyze the development of federated *Inter-Cloud* proposals, helping the unification of ideas and terminology by giving a holistic view of them, regardless of specific protocols and technologies.

The *Reference Architecture* is presented by using three architectural patterns. The first one describes the main components of federated *Inter-Cloud* systems; second, a pattern for an *Inter-Cloud Broker* is used for achieve the communication between the system components; and finally, an *Inter-Cloud Exchange* pattern shows how to perform the registration of *Service Provider and* services. By having an *Inter-Cloud Reference Architecture*, *Security patterns* and *Misuse patterns* could be used for securing the system.

The proposed *Patterns* and *Reference Architectures* are validated by verifying their correctness, completeness and coverage of specific architectures, and in the particular case of this thesis, by being also reviewed by experts in the field of patterns and computing systems, and published in several

international conferences. The individual papers were strongly discussed with the purpose of improving them before publishing them.

Important preliminary results have been achieved, three architectural patterns have been developed for describing and defining the *Inter-Cloud Reference Architecture* and also three *Misuse Patterns* were expressed and described using the *Inter-Cloud Reference Architecture* as a basis. This would allow us to follow known methodologies in order to achieve a *Security Inter-Cloud Reference Architecture*.

RESUMEN

Los sistemas *Inter-Cloud* son la evolución de los sistemas de *Cloud Computing* tradicionales; permitiendo la cooperación y adición de varios servicios provenientes de múltiples *Proveedores de Servicios* heterogéneos. Un caso especial de *Inter-Cloud* tiene como propósito la compartición de recursos entre *Proveedores de Servicios* otorgando el aprovisionamiento de recursos de manera inmediata, oportuna y escalable, alcanzando objetivos de *Calidad de Servicio* bajo condiciones variables. La cooperación entre ellos se logra formando una federación entre los *Proveedores de Servicios*; este tipo especial de *Inter-Cloud* es llamado *Inter-Cloud Federado.*

Actualmente existe mucha divergencia relacionada al entorno *Inter-Cloud.* Se han propuesto varios componentes, proyectos y arquitecturas, algunos de ellos refiriéndose a la misma solución, pero con distintos nombres, confundiendo y retrasando el avance de estas tecnologías y futuros avances relacionados a esta área. Estos problemas afectan también el proceso de asegurar el sistema.

Esta tesis propone la creación de una *Arquitectura de Referencia* para sistemas de *Inter-Cloud* federados utilizando patrones arquitectónicos y también, usar esta *Arquitectura de Referencia* como base para el proceso de aseguramiento de este tipo de sistemas.

Una *Arquitectura de Referencia* es una arquitectura de software genérica y estandarizada en un dominio particular, independiente de la plataforma. Ésta debe definir los componentes fundamentales de un sistema y la interacción entre estas unidades, siendo una herramienta útil para la construcción y entendimiento de complejos sistemas tales como el *Inter-Cloud.* Ésta catalizará el desarrollo de las propuestas de *Sistemas Inter-Cloud* federados, ayudando a la unificación de ideas y terminologías, dando una visión holística de ellas, sin tener en cuenta protocolos o tecnologías específicas.

La *Arquitectura de Referencia* es presentada utilizando tres patrones arquitectónicos. El primero describe los principales componentes de los sistemas de *Inter-Cloud* federados; el segundo, un patrón para un *Broker* es utilizado para lograr la comunicación entre los componentes del sistema; y finalmente, un patrón llamado *Inter-Cloud Exchange* muestra cómo realizar el registro de los *Proveedores de Servicio* y los servicios. Teniendo una *Arquitectura de Referencia Inter-Cloud*, se podrían emplear *Patrones de Seguridad y Patrones de Uso Indebido* para asegurar el sistema.

Los *Patrones* y *Arquitectura de Referencia* propuesta son validados al verificar su exactitud, completitud y cobertura sobre arquitecturas específicas, en el caso particular de esta tesis, siendo también revisados por expertos en el campo de los patrones y sistemas de computación,

presentándolos en varias conferencias internacionales. En ellas, cada *paper* fue fuertemente discutido con el propósito de mejorarlo antes de publicarlo.

Se lograron importantes resultados preliminares, desarrollando tres patrones arquitectónicos que describen y definen la *Arquitectura de Referencia Inter-Cloud* y también fueron representados y descritos tres *Patrones de Uso Indebido* utilizando como base la *Arquitectura de Referencia Inter-Cloud*. Esto nos permitiría seguir metodologías ya conocidas con el propósito de desarrollar una *Arquitectura de Referencia Inter-Cloud de Seguridad.*

1.1.1 INDEX

FIGURES

Chapter I

1 Introduction

1.1 General context

The vision of computing as a utility, where consumers pay only for what they use exists since the 60s; however, just a few years ago with the advent of *Cloud Computing*, this vision is provided virtually and remotely. This technology provides clients (or consumers) with various computing services without the need to acquire technological infrastructure and let them pay only for the amount of services they use, freeing them from expensive technology purchases and maintenance. All these problems are delegated to the supplier, who worries about the continuous delivery of services. *Cloud Computing* caused an increase in demand for such services, because it is offered to the consumer an apparently infinite amount of resources at a low price with a completely outsourced management service. However, this seemingly endless source of computing resources is tied to the *Service Provider* (*SP*) size, and there are only a few providers that actually can provide huge amounts of resources. *SPs* must ensure that there are sufficient resources available to them in case that the demand increases unexpectedly. Moreover, it is very common that resources are underutilized much of the time, a situation that is even worse when the supplier is large. This is caused by a not distributed uniformly demand for computational resources; there are peaks, which may be concentrated in the same time range, causing therefore resource exhaustion at some times, and a waste of computer processing capacity in others. Another problem is that consumers may need a service not supplied by their current provider.

As an answer to the above problems, federated *Cloud Computing* emerged, better known under the name of federated *Inter-Cloud* [Gro12, Buy10], and that refers to the agreement between *SPs* to share resources in order to increase their computational resources and provide a larger variety of services. The research community and some private organizations are beginning to develop architectures, technologies, and standards to support the integration of multiple cloud systems.

1.2 The problem

The increasing divergence and lack of a common guide in the *Inter-Cloud* environment slows down the development process, its adoption and the way to make it secure. Several components, architectures and projects have been proposed, some of them referring to the same solutions but with

different names; confusing and therefore delaying the advance of these technologies and further advances related to this area.

As it was mentioned above, the absence of a *Reference Architecture* makes difficult develop and to pose strategies that makes the system more secure. Without a clear guide that unify concepts and components, the securing process will be always related to specific technologies, making them too restricted, less customizable and poorly extensible.

1.3 Motivation

There are two main motivations in this work. The first one is the clear lack of guidance that could catalyze the development of the *Inter-Cloud* systems. The second motivation, is that due the absence of the previously mentioned guide, it is too difficult to be able to secure the system. So, proposing a well-defined guide for an *Inter-Cloud* system, we will be helping into the development of current and future *Inter-Cloud* projects, and also will establish the basis to perform a secure *Inter-Cloud* proposal.

As far as our knowledge, there is none previous guide or security references not related to specific tools or protocols. This would be the first proposal in that direction.

1.4 Proposal

Motivated by the current lack of clear guidance for approaching the field of *Inter-Cloud,* the goal of this work is to abstract a *Reference Architecture* from the current proposals; the *Inter-Cloud Reference Architecture* will also contribute into the securing process of the *Inter-Cloud* systems.

This *Reference Architecture* has the objective to be a guide to catalyze the development of *Inter-Cloud* systems, set common terms and components in order to improve the way different proposals refers to them, also to be the basis over the construction of a secure *Inter-Cloud* system. The *Reference Architecture* shall also be suitable to serve as a basis for thinking and communicating about *Inter-Cloud* developments and for giving some decision guidelines for architecting them.

By the way, into the process of building a secure *Inter-Cloud* system, *Security Patterns* and *Misuse Patterns* [Fer13a] seem to be good approaches to reach this target; but, before to apply these techniques and methodologies is necessary to have a well-defined *Reference Architecture* of the system to be secured.

This thesis proposes the use of patterns in order to abstract a federated *Inter-Cloud Reference Architecture* from current proposals, and use the same into the process of securing the system. Patterns are tools of great value because they provide a better understanding of the functional aspects

and can be complemented with other related patterns to achieve a more understandable and complete architecture.

In this work, six *Inter-Cloud* patterns are presented, three architectural patterns: the *federated Inter-Cloud* pattern, the *Inter-Cloud Broker* pattern and the *Inter-Cloud Exchange* pattern, which constitute the basis of the *Inter-Cloud Reference Architecture* proposal, and also, three *Misuse Patterns*: the *Steal information using a false service misuse* pattern, and two kinds of *Denial-of-Service in federated Inter-Cloud* patterns (*Exhaust resources through many request* and *Disrupt the search of services*), those will be used in future works to validate the security of the system (using the current *Inter-Cloud Reference Architecture* to present them). These patterns are presented using the POSA template [Bus96] and UML for modeling relations and interactions between components.

1.5 Hypothesis and Objectives

1.5.1 Hypothesis statement

This thesis proposes:

> *"A Reference Architecture for Federated Inter-Cloud systems could be abstracted from current proposals, capturing their main structure and behavioral aspects, and using them to express Security and Misuse Patterns"*

1.5.2 Objectives

The proposed work aims to define a *Reference Architecture* for *federated Inter-Cloud* systems that will be used to achieve a *Security Inter-Cloud Reference Architecture*. To achieve this, four major research works have been done:

- To perform an analysis directed to identify actors, components, functions, relationships, requirements and constrains of *Inter-Cloud* proposals.
- To identify patterns using current *Inter-Cloud* proposals and describe them based on the previous proposals.
- Create specific patterns for *federated Inter-Cloud* systems using the results of the previous objectives and using them to propose a *Reference Architecture* for *federated Inter-Cloud* systems.
- To describe *Security* and *Misuse Patterns* using the *Inter-Cloud Reference Architecture*.

1.6 Validation

Reference Architectures are not implementable; they are abstract models and cannot be evaluated with respect to security or performance through experimentation or testing. A *Reference Architecture* is similar to a pattern and it has a similar use, it is a paradigm to guide implementation of new systems or evaluation of existing systems.

Their evaluation is based on how well they represent the relevant concepts of the systems they describe, how well they handle abstract threats, how complete they are, how precise they are, how they can be applied to the design or evaluation of systems, and how useful they are for other relevant functions. Their final validation comes from experts and practitioners who can find them useful and convenient to build concrete architectures.

In the particular case of this thesis work, validation is also achieved by being reviewed by experts in the field of patterns and computing systems, being accepted, presented and published in several international conferences. The individual papers were strongly discussed with the purpose of improve them before publish them.

1.7 Methodology

Explicit the methodology used into the construction of this thesis work is necessary in order to be able to understand better this work and make it reproducible for others.

This thesis is the result of the following steps:

1. Capturing and studying the state of the art of the *Inter-Cloud* systems.
2. Concepts, actors, components and functions were unified and related.
3. Patterns that generalize inherent problems of the *Inter-Cloud* system were found, and then they were refined to achieve detailed patterns applicable to the *Inter-Cloud* systems studied.
4. Three architectural patterns were developed to define the *Reference Architecture* of the system, their components and responsibilities.
5. Three *Misuse Patterns* were developed using the *Reference Architecture* previously developed.
6. The proposed architecture and the security application of the same were also validated by being accepted, presented and published in several international conferences wherein pattern and computer science experts reviewed them.

During the development of this thesis, three scientific papers were published in international conferences. The details of each paper and conference name follows:

- O. Encina, E. B. Fernandez and R. Monge. "A misuse pattern for Denial-of-Service in federated Inter-Clouds". *Proceedings of 3rd Asian Conference on Pattern Languages of Programs*, Tokyo, Japan, March 2014.
- O. Encina, E. B. Fernandez and R. Monge, "Towards Secure Inter-Cloud Architectures". *Proceedings of Nordic pattern conference on Pattern Languages of Programs,* Sagadi Manor, Estonia, April 2014.
- O. Encina, E. B. Fernandez and R. Monge. "Threat analysis and misuse patterns of federated Inter-Cloud systems". *Proceedings of 19th European Conference on Pattern Languages of Programs*, Bavaria, Germany, July 2014.

The papers describe parts from the proposal, development and future works of this thesis.

1.8 Document structure

Chapter 2 presents to the reader background information that will be useful to understand better this work. Chapter 3 presents the state of the art used as reference to create and relate the proposed patterns. Chapter 4 presents the *federated Inter-Cloud Reference Architecture* by showing a system analysis and three architectural patterns: the *federated Inter-Cloud* pattern, the *Inter-Cloud Broker* and the *Inter-Cloud Exchange* pattern. Chapter 5 presents an on-going research and establishes the basis for developing a *Security Inter-Cloud Reference Architecture* by introducing and presenting three M*isuse Patterns*. Chapter 6 Discuss about the results achieved in this work and the possibilities to extend it. Chapter 7 presents Conclusions and future works.

Chapter II

2 Conceptual Framework

2.1 Cloud Computing

It was 1969 when the chief scientist of the ARPANET project, Leonard Kleinrock [Kle05] said:

> *"As of now, computer networks are still in their infancy, but as they grow up and become sophisticated, we will probably see the spread of 'computer utilities' which, like present electric and telephone utilities, will service individual homes and offices across the country."*

The computing technology has suffered a big and massive transformation; today computing services are available on demand as any other service available in the actual society. This means that consumers only need to pay providers when they use the computing services, avoiding the investment and maintenance of the *IT* infrastructure, tasks that are delegated to the *SPs*.

In this new service model, consumers access their services based on their requirements without regard to where or how they are delivered. This model fits well in the 'computer utility' idea, and is known under the name of *utility computing* or *Cloud Computing*.

The *National Institute of Standards and Technology* (*NIST*) defines *Cloud Computing* [NIS13] as:

> *"a pay-per-use model for enabling available, convenient and on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."*

On-demand self-service means that consumers are able to obtain without human interaction computing capabilities from a *SP*. *Broad network access* refers to the need of a network-based access and standardized mechanisms in order to facilitate access through heterogeneous platforms like desktop computers, tablets, or smartphones. *Resource pooling* describes provider's resources as a pool of different physical and virtual resources dynamically assigned to consumers. The assignment and reassignment are based on a multi-tenant model in order to achieve high utilization (usually using virtualization technologies). The possibility of rapid scaling up or down of provisioned capabilities is referred to as *rapid elasticity*. Moreover, the impression of infinite resource capabilities that are

obtainable in a large quantity at any time is proposed. Finally, *measured service* means resources are automatically monitored, controlled, and if needed, optimized using metering mechanisms appropriate to the resource type. Furthermore, the according utilization is transparently auditable for consumers and *SPs*.

SPs offers infrastructure, platform, and software (applications) as services, which are made available as subscription-based services in a pay-as-you-go model to consumers; such services are known as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). They differs in the complexity of the provisioned capability, and how customers can access or use them. IaaS is about the provision of basic computing capabilities for customers like virtual machine instantiation. In PaaS, user have access to development platforms like data bases, and to a certain extent the configuration of hosting environments. In the SaaS model, customers just use a certain application running on a cloud infrastructure.

The *Cloud Computing* aims to join data centers in a kind of network of virtual services, where users can access services anywhere in the world under demand at competitive costs depending of the consumer requirements. Today, developers and entrepreneurs will no longer require large investments in hardware or platforms to deploy their business. *Cloud Computing* frees them from low level tasks like worry about the hardware or software infrastructure, focusing them only in their business.

2.1.1 Current Cloud Computing limitations

Cloud Computing caused an increase in demand for such services, because the customer is offered an apparent infinite amount of resources at an inexpensive price with a completely outsourced management service. However, this seemingly endless sources of computational resources is tied to the size of the *SPs*, and there are only a few providers that actually can provide huge amounts of resources. *SPs* must ensure that there are sufficient resources available to them in case that demand is increased unexpectedly. It is known that in small suppliers resources are underutilized much of the time, a situation that is even worse when the supplier is larger. This is caused because the demand for computational resources is not distributed uniformly, they have peaks concentrated in the same time range, causing therefore resource exhaustion at some times, and a waste of computer processing capacity in others.

In order to improve the service delivery, *SPs* have setup several data centers at different geographical locations over the Internet in order to optimally serve needs of their customers around the world. However, existing systems do not support mechanisms and policies for dynamically coordinating load distribution among different Cloud-based data centers in order to determine optimal location for

hosting application services to achieve reasonable QoS levels. Further, the *Cloud Computing SPs* are unable to predict geographic distribution of users consuming their services, hence the load coordination must happen automatically, and distribution of services must change in response to changes in the load.

By the way, big *SPs* like Amazon Web Services [1](AWS), Rackspace[2] or Windows Azure[3] use proprietary middleware, isolating their technologies and environments. This isolation slow down the development of *Cloud Computing* technology. The aforementioned issues lead to many disadvantages for consumers, such as vendor lock-in and high migration costs (in some cases they can't even migrate) due the lack of interoperation.

Other common problem is that consumers may need a service not supplied by their current provider; it that situation, the consumer can quit their *SP* and migrate to other one, or resign from having such service.

2.2 Inter-Cloud

In response to the aforementioned problems, several standardization efforts and ideas have been taken place. Protocols for information exchange and common interfaces are crucial in order to facilitate the interoperability.

On the other hand, it exists the idea of having *SPs* that could share their resources. Allowing the diversification and the amount of offered services will also help to foster the development of small and medium size providers that can associate and compete with big *SPs*, creating a more competitive marketplace for *Cloud Computing* services.

That is why the latest trend in *Cloud Computing* research is dominated by the idea of federating heterogeneous clouds. Considering a new *Inter-Cloud* scenario where different clouds, belonging to different administrative domains, can interact with each other, sharing and gaining access to more and different services, increasing their computational and storage capabilities, and becoming themselves (and at the same time) in consumers and *SPs*.

Federated *Cloud Computing* environment or *Inter-Cloud* envision just-in-time, opportunistic, and scalable provisioning of application services, consistently achieving QoS targets under variable workload, resource and network conditions. The overall goal is to create a computing environment

[1] aws.amazon.com
[2] http://www.rackspace.com
[3] http://azure.microsoft.com/

that supports dynamic expansion or contraction of capabilities (VMs, services, storage, and database) for handling sudden variations in service demands.

Several benefits will arise from the cloud federation covering from business/economy to technical issues that are important to stakeholders like consumers, developers and *SPs*. From a business/economy perspective, the advantages to consumers are the non-existence of a vendor lock-in, a consumer will not be any more restricted to a single *SPs*. Developers and *SPs* are going to be able to distribute and offer cloud applications across several *SPs*. Consumers are going to experiment a reduction in cost due the fact that moving services to a different and more adequate *SPs* will be easier.

From the technical point of view, developers will have several advantages. The monitoring of a distributed infrastructure scaled over several clouds will be possible thanks to the interoperability. Also, a better response to the QoS offered due the possibility of automatically reacting over one violation in the agreements. The services could be verified and the discovery of services will be secure if proper security measurement are applied to the *Inter-Cloud Broker* and *Inter-Cloud Exchange* components [Enc14d, Enc14e]. Due *Inter-Cloud* could scale over several clouds around the world, fault tolerance and an inherent disaster recovery system could be implemented (the information will be distributed and replicated).

The above advantages gives to cloud providers a strong incentive for achieving interoperability. One strong motivation for cloud providers is an increase in market share, as interoperability further drives wide-spread adoption by users and developers in using *Cloud Computing*. This translates to a growth in revenue. Another factor is to be able to provide users with a better SLA guarantee, in case of hardware failure and/or security attacks.

Although there are many motivations for creating such cloud interconnections through federation, there are some challenges that must be resolved before.

2.2.1 Inter-Cloud challenges

The *Inter-Cloud* computing is a new area and it's in an early development stage, several research challenges must be tackled, some of them are summarized below:

- Automatic Response and Matching: Each *SP* that request a service using discovery mechanisms should be able to receive the best possible service available according to its requirements; it must also be able to react to unexpected changes in its provider and also in the whole *Inter-Cloud*.

- Resource Management: *SPs* must keep a complete control over its resources, and known exactly the amount of resources available for its own needs and the amount of resources that can be shared with others. *SPs* should not decide to share its resources based in instantaneous requests, they also have to consider a mechanism that could predict its own future behavior and estimate the amount of resources that are actually available to others. The decision must be taken considering more factors and the historical behavior.
- Resource migration: The need to be adaptable require of being able to easily migrate some resource to another host. For example, virtual machines could be migrated in order to provide a fault-tolerant system independently from the virtualization format used in the destination place.
- Monitoring: Both, *Services Providers* client and providers must be able to monitoring its resources in order to comply or demand the agreement compliance between *SPs*. The monitoring system should be scalable, in the same way that *SPs* are.
- Agreements Compliance and QoS: The *Inter-Cloud* needs to provide mechanisms for facilitating the managing of agreements compliance between *SPs*; the most popular solution proposed is the use of Service Level Agreement (SLA) as mean of contract.
- Security: It is a major concern for cloud users and *SPs*; an *Inter-Cloud* could only create new threats. The primary concern is that tasks will cross from one administrative domain to another, and sensitive information about the tasks and users could be disclosed or tampered during this migration. A seamless migration of tasks in an *Inter-Cloud* requires a well thought about trust model.
- Policy Management: *SPs* must respect and adopt their policies and rules. For example, it is needed to integrate different security technologies, allowing *SPs* to be able to join the *Inter-Cloud* without changing their security policies.
- Naming: In the *Inter-Cloud*, resources could be distributed in several clouds. It is necessary to identify uniquely every resource, creating different name spaces and naming systems. Naming is related to Identity Management which is a more general mechanism.

2.3 Inter-Cloud standards

In April 2011, IEEE announced the raising of two working groups (i.e., P2301, P2302) aiming to define portability and cooperation mechanisms among different clouds [Cal12].

The *IEEE P2302 Intercloud Working Group* published a draft *Standard on Intercloud Interoperability and Federation* (*SIIF*) [SIIF] that proposes an architecture that defines topology, functions, and governance for cloud-to-cloud interoperability and federation [Dem12]. Topological

elements include clouds, roots, exchanges (which mediate the governance between clouds), and gateways (which mediate data exchange between clouds). Functional elements include name spaces, presence, messaging, resource ontologies (including standardized units of measurement), and trust infrastructure. Governance elements include registration, geo-independence, trust anchor, and potentially compliance and audit.

The IEEE P2302 SIIF architecture is originated from the paper published in 2009 by Cisco [Ber09] that tried to leverage the basic routing and messaging Internet protocols such as BGP, OSPF, XMPP to address Intercloud integration and interoperability [Dem12].

In [Wan12] is discussed the interoperability among clouds about message transmission, proposing the Extensible Messaging and Presence Protocol (XMPP) as the protocol that best suits this task; data transmission, proposing the Uniform Data Format (UDF) to the exchange among different storage solution engines, and virtual machine transfer, taking the [SIIF] project as a reference.

2.4 Reference Architectures

A *Reference Architecture* (*RA)* is a standardized, generic software architecture, in a particular domain and with no platform dependencies [Fer13b]. This should define the fundamental components of a system and the interaction among these units, being a useful tool for building and understanding complex systems such as *Inter-Clouds*. It provides a guide for the development of architectures for concrete versions of a system or to extend it. It is created by capturing the essentials of existing architectures and by taking into account future needs and opportunities, for a whole variety of specific technologies, patterns and business models. It can also be derived from domain models.

The purpose of a *RA* is to provide guidance for the development of new versions of architectures for systems [Ree02, Mul07].
There are several benefits and uses of a reference architecture:

- **Provides a modular and flexible technology base**: The aim of the reference architecture is to enable the development of more flexible solutions, in which individual components can be added or replaced more easily than in a solution without it.
- **A way to understand complex systems such as an Inter-Cloud**: They are more effective than textual descriptions or imprecise block diagrams (e.g. it does not provides information about the cardinality of relations).

- **A basis for concrete Inter-Cloud implementations**: *RA* oriented to specific products can be derived from another *RA*. A new vendor of *Inter-Cloud* systems can use the *Inter-Cloud RA* to define its own product.
- **Enables the faster deployment of new technology**: Due that the components are independent from each other, it is easier to deploy new components as long the new ones maintain compatibility with existing interfaces. The replacement or alteration of one is still simpler.
- **A way to unify Inter-cloud terminology**: Different vendors have different ways to describe their services and products. A *RA* can be used as a framework to unify terms and descriptions. This is useful for comparing and selecting different *Inter-Cloud* systems.
- **Enforcement of standards and regulations:** A *RA* can be used to support standards and regulations, which can be described as policies, and in turn can be implemented as patterns and become part of the *RA*. It helps architects or designers to identify what components of the cloud system are associated with the standard and can be used to comply with the specific rules of the standard. Applications derived from the *RA* will easier comply with the standards or regulations [Fer14b].
- **Service certification:** Critical applications require the use of certified services. Even in non-critical applications, certification increases the trust of the consumer. Certification approaches may use ontologies or other formal models to describe certificates as well as monitoring requirements. A *RA* can be used to guide the certification process. An *Inter-Cloud* system can show that its services can handle the corresponding requirement (e.g. like availability, which can increase customer trust).
- **Reference for monitoring functions:** Monitoring requires mechanisms to obtain information about the system status. A *RA* provides guidelines about the places where events should be collected in order to fulfill SLAs requirements and for system administration. It may provide a guide for distribution of monitoring functions to make them more auditable.

Although *RA*s don't have a formal definition, [Avg03] presents a list of elements that they should include:

- The system stakeholders that interact with the system such as customers, administrators, developers and IT staff.
- The views described in a *Rational Unified Process* (*RUP)* like*:* use case model, analysis model, design model, deployment model, and implementation model. These views should be developed using *Unified Modeling Language* (*UML*).

- The architectural patterns that characterize parts of the architecture.
- The quality attributes that are desirable for the system and must be supported, it will suffice to mention the quality attributes that are important for this specific category of systems, and therefore should be supported; for example: interoperability, modifiability, portability, usability, reusability, and integrability.

A *RA* should be described at an abstract level, and all details about implementation should only be considered for concrete architecture.

In this thesis, patterns are used to specify the relevant parts of the *RA*.

2.5 Patterns

Patterns are encapsulated solutions to recurrent problems and define a way to express requirements and solutions concisely, as well as providing a communication vocabulary for designers [Bus96, Gam94]. The description of architectures using patterns makes them easier to understand, provides guidelines for design and analysis, and can define a way of making their structure more concise, expandable and reusable [Fer13a].
Patterns express a relation between a certain environment (context), a problem (conflict of forces) and a solution (resolution of the conflict). They describe recurrent designs on a medium level of abstraction. They rarely exist in isolation; each pattern generally has one or more relationships with other patterns [BHS07b, Ris98]. A collection of patterns can be collected in a pattern language for a particular purpose such as presenting patterns within a particular architecture, so that it can be organized into problem areas that characterize the architecture [BHS07b].

Typically patterns provides a solution using UML diagrams such as class diagrams (static) and sequence diagrams (dynamic). These diagrams present a precise way of describing a system allowing designers to use them as guidelines. A pattern is described using a template. The POSA template [Bus96] is followed for this work, which consists of the following components: intent, context, problem, solution, implementation, known uses, consequences, and related patterns.

Patterns can be used to build RAs, which we do in this work.

2.6 Security Patterns

Security Patterns are very useful tools to design secure systems, they encapsulate defenses to threats in a concise and reusable way; catalogs of them have been built in [Fer13a]. This kind of patterns describe solutions to the problem of controlling (stopping or mitigating) a set of specific threats through some security mechanism, defined in a given context [Fer13a]. This solution needs to resolve

a set of forces. Furthermore, a set of consequences indicate how well these forces were satisfied; in particular, how well the attacks were handled. The most common use of *Security Patterns* is to help application developers -who are not security experts- to add security in their designs. Patterns of this kind also are used to reinforce a legacy system.

2.7 Misuse Patterns

In order to design a secure system, it is necessary to understand possible threats to our system. Several methods have been developed to identify threats [Bra08, Red07]. Once identified, it is necessary to describe how these threats are performed (what units it uses and how) to achieve a misuse according to the goals of the attacker. A misuse pattern describes how a misuse is performed from the point of view of the attacker [Fer13a]. It defines the environment where the attack is performed, countermeasures to stop it, and it provides forensic information in order to trace the attack once it happens.

Misuse Patterns are useful for developers, because once they determine that a possible attack can happen in the environment; a corresponding misuse pattern will indicate what security mechanisms are needed as countermeasures, analyzing ways of stopping the attack by enumerating possible *security patterns* that can be applied for this purpose. Also, misuse patterns can be very useful for forensic examiners to determine how an attack is performed, and where they can find useful evidence information after the attack is done.

An important value of misuse patterns is that they describe the components of the system where the attack is performed using class diagrams and sequence diagrams, relating the attack to specific system units. A catalog of *Misuse Patterns* is needed to let designers evaluate their designs with respect to possible threats.

Chapter 3

3 State of the art

3.1 Inter-Cloud projects

The *Inter-Cloud* environment has seen several projects [Ber11, Buy10, Car12, Gro12, Meh10, Kec12a, Ker12] that aim to solve the lack of federation and interoperability capabilities between heterogeneous *SPs*; however, the lack of a *Reference Architecture* has led to some of them to propose similar but good solutions worthy of being considered. Due to these commonalities, we have been able to identify a federated *Inter-cloud* pattern which relates the most popular federated *Inter-Cloud* projects in order to conceptually unify them and to build the basis of an ongoing research about secure *Inter-Cloud* architectures.

[Gro12] presents a complete survey of brokering architectures and existing applications, which reviews various *Inter-Cloud* projects, making a distinction between federated systems (federated *Inter-Cloud*) and non-federated ones called *MultiClouds* (multiple independent Clouds which do not share resources). While [Gro12] reviews many relevant projects and provides a classification of them, it does not consider architectural aspects of *Inter-Cloud* proposals and their security aspects, focusing more in current projects and classifying them, than in unify a terminology and common components as we do in this work.

There are several proposals for *federated Inter-Cloud* systems. First, *InterCloud* [Buy10] proposes a centralized Marketplace around some *Cloud Coordinators* and *Cloud Brokers* which receive and redirect the requests and return responses from and to the Marketplace. An *SLA* specifies the details of the service to be provided in terms of a metric agreed upon by all parties, and incentives and penalties for meeting and violating the agreements, respectively [Buy10]. This work is one of the first proposing a federation for resource sharing between *Clouds* and presents a concrete system in that sense; although the architecture and components proposed are similar in terminology and functions with our work, differs from our proposal because the authors did not care about security issues; they propose challenges that justify the existence of an *InterCloud* and also performs an experiment using *CloudSim* [Buy09b] to evaluate the performance of the proposed system.

Second, the *Federated Cloud Management* project [Kec12a] proposed a single point of access to the federation called *Generic Meta Broker Service*. This component takes care of the submission of a

request from the Consumers and worries about matching it with the best possible provider. The *Federated Cloud Management* uses an external component, a database (the *Federated Cloud Management repository*) and internals caches (*Native repository*) which improve the performance of the system by avoiding virtual images to be constantly moving in the system because the cache saves them for possible reuse. The *Generic Meta Broker Service* uses external *Cloud Brokers* for instantiation and to redirect the calls to the *SPs*. In [Kec12a] an extra component is included to improve the way the *Generic Meta Broker Service* makes decisions and matching (the *Global Autonomous Manager*), by monitoring and analyzing the information arrived from the *Cloud Brokers*. This work is mainly focused in performance (metrics are used to choose between different *SPs*) and fault tolerance issues (replicating the *Native Repository*, for example), no references to security issues.

Third, in the Horizontal Dynamic Cloud Collaboration Platform [Meh10], the consumer interacts transparently with the *Virtual Organization based dynamic collaboration platform* (VO-based component). This system includes a *Catalog* inside the *SP* which helps to answer the requests coming directly from the consumer to their subscribed Cloud. The Clouds are federated forming a dynamic collaboration network for sharing resources using the VO-based component to resolve and assign the best combination of *SPs*. Unlike our work, there is no reference to any security issue in this paper; however, it is possible to realize that is considered because the architecture of the *primary Cloud Provider* (*pCP*) includes a policy repository, and a kind of *Admission* controller that possibly correspond to a Access Control System that has still not be defined. This work also does a simulation that offer results justifying their dynamic collaboration approach.

Finally, the Contrail approach [Car12] is one of the most complete *Inter-Cloud* proposals focusing specially in authentication and authorization methods. This project uses a component called Federation Runtime Manager as a single front end for all services, dedicated to map user requests to cloud resources. The Contrail approach [Car12] uses multiple Federations Access Points distributed over the network, each one potentially in contact with all *SPs* in the federation. This project also offers the possibility (in contrast to the others) to federate or not resources using specials adapters in the federation access points, so this project works forming federated *Inter-Clouds* and Multiclouds.

The previous projects have been classified in [Gro12] as centralized Federated *Inter-Clouds*, and they were used to create the *federated Inter-Cloud* pattern.

In [Cel10a] is proposed the use of a component called *Cross-Cloud Federation Manager*, that allows to a cloud to establish a federation with others using a three-phase model: discovery, match-making

and authentication. The technologies on which such implementation rely are XMPP, XACML and SAML, respectively (for the design of the discovery agent, the match-making agent and the authentication agent). [Cel10b] presents a reference architecture to address specifically the Identity Management (IdM) problem in an *Inter-Cloud* context, and also show how it could be applied to manage the authentication needed among clouds for the federation establishment. Unlike ours, this work emphasizes the use of known protocols and tools to resolve problems and propose an existing solution to the *Inter-Cloud*, although its proposal is very concrete, it is also limited to the scope of different architectures, protocols and tools; also, the main focus of this work is the *Identity Management* problem, that is not yet covered in our work but that is going to be addressed in future works when becomes necessary to apply *Security Patterns* to build a *Secure Inter-Cloud Reference Architecture*.

The three-phases model is also present in [Cal12], where are identified the main issues concerning the establishment of federation in a decentralized *Inter-cloud* environment, focusing on how the discovery and matchmaking phases takes place between cloud providers and the establishment of a federation. It presents centralized and decentralized brokerage approaches including cloud providers and brokers, detailing how cloud providers might find out partners for federation in a dynamic fashion. Four types of schemas are proposed and analyzed: centralized, hierarchical, decentralized and mixed. The main focus of this work is to solve the problem of finding partners to establish a federation by proposing solutions and comparing different architectures, arguing in favor of the decentralized architectures, but without giving any approach for security issues.

[Jra12] presents a generic architecture of a Cloud broker operating in an *Inter-cloud* environment. The broker aims to find the most suitable Cloud provider while satisfying the user's service requirements in terms of functional and non-functional Service Level Agreement (SLA) parameters; the paper describes the services that a broker should have and their design, focusing on the SLA management and resource interoperability. Finally, it presents a simulation of the proposed architecture by using the *CloudSim* tool described in [Buy09b]. This work has a strong focus in the use of SLA for the Discovery, Match-making and Monitoring phase; our work does not contain any special or specific technology like the SLA agreements, making this work more specific but also more static and less generic.

3.2 Inter-Cloud Security

Some works has been very detailed in proposing trust, identity management, authorization, and authentication solutions for securing the *Inter-Cloud* [Ber10, Ngo12, Sin13]; however, these security

approaches are also mixed with special components, technologies and long descriptions of protocols and specifications, which make their proposals cumbersome and hard to implement in legacy designs or in ongoing *Inter-Cloud* developments. No effort to define a holistic view of the federated *Inter-Cloud* system has been proposed.

[Ber10] proposes the new Intercloud *Trust Model* in conjunction with the prevalent PKI based *Trust Model*, stating that current PKI certificates based trust model is unsuitable for an Intercloud environment. According to the current PKI based trust model, once a CA authorizes a certificate for an entity, the entity is either trusted or non-trusted. A *Trust Index* is proposed as a complement. The *Trust Index* is the level of trust demonstrated by cloud providers; depending on the level of trust an *Intercloud* provider might trust another provider or not; for example, storage services, but not to execute programs using that services. The *Trust level* is specified within a given time (the trust level today is not necessary the same as a year ago). The *Trust level* information is dynamic, in opposition to static PKI certificates. The paper ([Ber10]) also discussed security related governance considerations within an Inter-cloud computing environment stating several problems with this topic. For example, to use the execution or storage of information beyond the boundaries (laws) of our country could lead that another entity (country) can access the information located under its domain; or also, the provider could retain the data for some reason, or even run mining algorithms over the data for secondary uses, such as for market research purposes. Unlike our work, this one presents several important issues related to the Trust index, but also performs a review of security governance problems. Trust issues will be addressed in future works (when applying security patterns). Moreover, this work propose an *Inter-Cloud* topology that fits with our *Reference Architecture*.

[Ngo12] attempts to solve the problem of providing an effective and robust authorization mechanism in which an entity can delegate permissions to another party to access its data in the *Inter-cloud* environment; for that purpose is proposed a dynamic trust establishment protocol for distributed authorization. Based on the attribute-based access control model (ABAC). [Sin13] presents a proxy-based multi-cloud computing framework, addressing subjects about trust, policy and privacy issues in an environment without pre-established collaboration agreements or standardized interfaces. This work mention specific security issues associated with collaboration among heterogeneous clouds that should be solved for any *Inter-cloud* proposal, such as: the establishment of trust among different cloud providers to encourage collaboration; to address the policy heterogeneity among multiple clouds so that composite services will include effective monitoring of policy anomalies to minimize security breaches and maintain the privacy of data and identity during collaboration. Both previous works address specific security issues, the first one ([Ngo12]), tackling only the trust establishment

problem, and the last one ([Sin13]), presents an overview about of trust, policies and privacy concerns; none of them provides a concrete architecture or gives an overview of the components that should interact between each other.

In [Boh13] work, it is stated the reduction of the risk for data and applications in public clouds by the use of multiple and simultaneous clouds. It provides four distinct models in form of abstracted multi-cloud architectures. The basic underlying idea is to use multiple distinct clouds at the same time to mitigate the risk of malicious data manipulation, disclosure, and process tampering. Four architectural patterns are distinguished:

- Replication of applications, enabling the user to get an evidence on the integrity of the results, answering the question: How does a cloud customer know whether his data were processed correctly within the cloud?
- Partition of application's Systems into tiers, separating its logic from the data, giving additional protection against data leakage due to flaws in the application logic. It answer the question: How a cloud user can be sure that the data access is well implemented and enforced effectively, and that errors in the application logic do not affect the user data?
- Partition of application logic into fragments; so, no cloud provider learns the complete application logic, and no cloud provider knows the calculated results of the application, giving confidentiality to data and applications. It answer the question: How can a cloud user avoid fully revealing his data or processing logic to the cloud provider?).
- Partition of application data into fragments; so, none of the involved cloud provider gains access to all the data, which safeguards data confidentiality.

The above work ([Boh13]) follows similar methodologies to extract from several works solutions that are presented as patterns, however; it does not follow any pattern template for documenting them. Also, the patterns presented comes from multiple general solutions in security when is given the partition or division of work is needed, and not any particular Inter-Cloud proposal like in our work.

3.3 Cloud Computing Reference Architecture & Patterns

A *RA* was developed in [Fer13b] to have a precise view of cloud systems. Since clouds are complex systems, each service model was presented in the form of patterns which describe their requirements, characteristics, main units, and the relationship between these units. Some use cases were also included that describe common functions for cloud services in general as well as for each service model. Various reference architectures for *Cloud Computing* have been

proposed by different organizations, such as IBM[4], HP[5], NIST [NIS13], and Oracle[6], giving their proprietary solutions or not considering security aspects. Our work is based on the methodology of Fernandez ([Fer13b]), using patterns into the construction of the *RA*.

3.4 Towards a Security Inter-Cloud Reference Architecture

The security in *Inter-Cloud* systems is one of the most important challenges because of the need for dynamic sharing, federation, privacy and collaboration across multiple clouds. Security in this type of systems relies heavily on the establishment of trust among the involved components. Users trust a *SP* to run or store some service, but the boundaries of trust become larger than the normal case of *Cloud Computing* because that *SP* could share their client's processes/resources with another provider. The user might not even notice this sharing. Without a strong focus on security this kind of systems becomes in a very attractive target for attackers, then, compromising a federated cloud would mean enabling the access to a larger list of user's private data and resources.

A *Security Reference Architecture* (but for *Cloud Computing*) was proposed in [Fer13b] *(*a *Security Reference Architecture* is a *Reference Architecture* where security services have been added in appropriate places to provide some degree of security for the complete environment), including defensive measures to secure cloud environments combining both security and misuse patterns in order to add security features to the previous *Reference Architecture*. By checking if a threat (misuse pattern) can be stopped or mitigated in the *Security* reference architecture, its level of security is evaluated.

In the *Inter-Cloud* field, a *Security Inter-Cloud Reference Architecture* has not been proposed. The NIST published a report on a *Security Cloud Reference Architecture* [NIS13], which describes in great detail all the aspects of such architecture, but no reference to a *Federated Inter-Cloud* system has been described. This report is the most complete *Security Reference Architecture* published until now. In most of these architectures, the lack of more precise or rigorous models is a clear weakness. They typically use block diagrams, which show the involved components but not the way they are associate with each other. The fact that an association between components is one to many does not show in the models and it is not clear where the security mechanisms should be attached.

[4] www.ibm.com
[5] www.hp.com
[6] www.oracle.com

3.5 Summary

The current scene of the *Inter-Cloud* environment is characterized by two kinds of works, those related to architectural proposals and those that address security issues.

Architectural proposals are very concrete, presenting several actual technologies and tools that address federation and interoperation issues between heterogeneous *Cloud Computing*. However it is possible to distinguish two big problems within them. The first one is due the lack of a common terminology, the proposition of similar components and technologies using different names slow down the comprehension, advance and cooperation between them. The second problem is about the specific components they propose, while is very convenient and useful to known which technologies actually solves a kind of problem; it also must be complemented with a bigger perspective, that allow the cooperation and understanding, easing future development of components independently of the proposed protocols or tools, that contrary could only delay or stop the development of the Inter-Cloud environment.

The second kind of works about the *Inter-Cloud* are those related to security issues; in the same direction that architectural propositions, they presents security solutions based mostly in specific tools or protocols (i.e. SLA is by far the most proposed technology to address agreements between Clouds). There are also here two big problems: the first one, is the fact that exist the need to couple the security solution to the architectural solution, the difficulty that would imply performing that adjustment is high; and is possible that both cannot even be coupled due specific tools or protocols in the architecture or security solution; the second problem is related to the first one, due it will be very difficult to extend the security if the architecture or some specific technology are changed.

There is no proposal in the *Inter-Cloud* environment that cover in a big perspective and with a clear notation the relation and functions between components, unifying the terminology. The same guidance or proposal should also be useful to improve the way security is addressed independently of specific protocols and technologies.

Chapter 4 - Results

4 Inter-Cloud Reference Architecture

It can be harder to achieve security if a comprehensive and detailed architecture for building or evaluating this kind of systems is not developed; the lack of a common architecture, more than delay the process of *Inter-Cloud* adoption has a negative impact on the process of making these systems more secure. In order to approach such problems, an *Inter-cloud Reference Architecture* has been developed.

This chapter presents a *Reference Architecture* that was developed mainly from the abstraction of current federated *Inter-cloud* proposals. The architecture is defined first by analyzing its stakeholders, the use cases where they are participating, and a brief description of each one. Then, three patterns are presented; the first, shows the *Federated Inter-Cloud* pattern, their main components, benefits and disadvantages. In second place, the *Inter-cloud Broker* pattern is presented showing the way that the communication and interoperation is performed. Finally, the *Inter-cloud Exchange* pattern is presented showing how different *SPs* would register and find services in the *Inter-cloud* environment.

The patterns are presented using the POSA template and UML notation in order to describe more precisely the components of the reference architecture and their relations. Every pattern is also described using statics and dynamics diagrams of the most important use cases of the federated *Inter-cloud* operation.

4.1 Federated Inter-Cloud use cases

4.1.1 Stakeholders (actors)

To understand the system operation it is necessary to look at the actors which participate in an *Inter-Cloud* system. We identify here the stakeholders involved in the operation of the *Inter-Cloud*, which include:

- *Service Consumer* (customer)—this can be an individual or an institution collectively denoted as *Party*.
- *Service Provider*—it is a company or institution providing a complete set of cloud services, a level of services (e.g. SaaS), or a specific set of services (typically through a broker).
- *Inter-Cloud Broker* – it is an intermediary between different *SPs* and also is in contact with the *Inter-Cloud Exchange* that belongs to the federation.

- *Inter-Cloud Exchange* – Acts like a market where the requests from customers and offerings from *SPs* are exchanged or linked. It is also responsible for maintain the *Catalog* of services offered and their characteristics.

4.1.2 Use Cases

The use cases and their actors are listed below, it is important to notice that these are just a subset of them and only are listed those related to the federated *Inter-Cloud*.

4.1.2.1 Service Consumer- customer (Actor):

The use case of the *customer* represents all the interactions that relate *Service Consumers* with the *Cloud Federation* (although indirectly through the *SP* and the *Inter-*C*loud Broker*). The *consumer* never realizes about its participation in the *Cloud Federation,* it just requests resources to the *SP* and verifies the fulfillment of the agreements. The main use cases for this actor are*: Request service*, *Cancel Service,* and *Monitor Service*.

Although the *Inter-Cloud* is transparent for the final consumer, some use case are described below in order to let the reader understand better the interactions. The *Pay Bill* use case that is commonly presented in *Cloud Computing* analysis and always associated to the customer, is avoided here because real clients are the *SPs* that request services to the *Inter-cloud*. A description of the related use cases related are:

UC 1. **Request service**: A consumer requests a service to the *SP*. If the *SP* cannot provide the service due lack of resources or because it does not have it available, forwards the request to the *Inter-Cloud* system. Finally, the *SP* deploys the service based on the consumer's request.

UC 2. **Cancel service**: A consumer request to cancel a service that has been deployed through the *Inter-Cloud*. The request is forwarded to the *Inter-cloud* and has to be reflected in the *Inter-cloud Exchange* and the *SP* that is providing the requested service.

UC 3. **Monitor service**: A consumer request to monitor a service that is under execution. The consumer selects parameters to be monitored. Specific information about for example *QoS* parameters are extracted directly from the local machine where the service is running; Due to the *Inter-Cloud* is transparent for the consumer, more information about the service is managed directly by the *SP*.

4.1.2.2 Service Provider (Actor)

The use case of the *Service Provider (SP)* represents all the interactions that relate *SPs* with *Consumers* and the *Cloud Federation* directly through the *Inter-Cloud Broker*. The *SP* establish an agreement with the *Cloud Federation* in order to share resources with others *SPs*. The main use cases

for this actor are: *Join Federation*, *Leave Federation*, *Create Service, Provide Service*, *Request Service, Cancel Service*, *Monitor Service, Request agreement* (e.g. in *SLA* form) , *Publish Service* (offering) and *Remove Service*. A description of the use cases are:

UC 4. **Join Federation**: A *SP* requests through an *Inter-Cloud Portal* its participation in the cloud federation. The *Inter-Cloud Portal* sends the information to the *Inter-cloud Exchange* where the *SP* is finally registered.

UC 5. **Leave Federation**: A *SP* requests to leave cloud federation through the *Inter-Cloud Portal:* The *Inter-Cloud Portal* sends the information to the *Inter-Cloud Exchange* and updates the *Catalog* by removing the services of that provider.

UC 6. **Create Service**: The *SP* creates or instantiates a given service and then it is published in the *Inter-Cloud* by sending a request for publishing.

UC 7. **Provide Service**: The *SP* initiates the execution of the given service, the statistics about it are registered locally.

UC 8. **Request Service**: A request for service is received from the *SP*. If the *SP* cannot provide the service due to lack of resources or because it does not have it available, forwards the request to the *Inter-Cloud* system. Finally, the *SP* deploys the service based on the consumer's request and is then delivered to the consumer. Also, it is possible that the *SP* request a service by himself, the parameters are the same that in the case when a consumer request it.

UC 9. **Cancel Service**: The *SP* sends a request to the *Inter-Cloud* to stop a service in execution. The request goes through the *Inter-Cloud Broker* to the *Inter-Cloud Exchange* where the service is unbound.

UC 10. **Monitor Service**: Specific information about for example *QoS* parameters are extracted directly from the local machine where the service is running. Information about the provider giving the service is obtained requesting to the *Inter-cloud Exchange*.

UC 11. **Request agreement**: The *SP* request information about the agreement of a given service that is being or it was provided. The agreement information is obtained from the *Inter-Cloud Exchange*.

UC 12. **Publish Service**: The *SP* request to publish a service by giving specific information about the service and the conditions offered for the service (QoS compromises). The request is delivered to *the Inter-cloud Exchange* where the service is registered.

UC 13. **Remove Service**: The *SP* requests to remove a specific service from the Inter-cloud. The request is delivered to the *Inter-cloud Exchange* where the Catalog eliminates the information about the availability of the service.

4.1.2.3 Inter-cloud Broker (Actor)

The *Inter-Cloud Broker*'s aim is to communicate heterogeneous and distributed *SPs*, and connect them to the *Inter-Cloud Exchange*; for that reason, their main use cases are related to the forwarding of request between different *SPs* and between *SPs* and the *Inter-Cloud Exchange*. It is reasonable also to think about the *Inter-Cloud Broker* as a cache which manage local information about its *SPs*, but that cache property will not be investigated further in this work. The main use cases for the *Inter-Cloud Broker* are: *Forward requests* (request for service, for search, for registering, etc.). A description of related use cases related are:

UC 14. **Forward request for service**: The request for service is launched in two occasions. First, when a *SP* needs to ask for a service to the *Inter-cloud* specifying what service is needed and some constraints about it. The request is forwarded to the *Inter-Cloud Exchange* through the *Inter-Cloud Broker*. And second, the request is launched by the Inter-Cloud Exchange itself after the matching process is complete. The *Inter-Cloud Exchange* creates a request for service and delivers it to the *Inter-Cloud Broker* that forwards it to the selected *SP*.

UC 15. **Forward request for search:** A request for search is launched by the *Inter-Cloud Exchange* once a request for service arrives. The request for search is delivered to the *Catalog* component where the information of the searched service is looked for.

UC 16. **Forward request for registering:** When a request for register is launched by a SP, the request is forwarded to the *Inter-Cloud Exchange* through the *Inter-Cloud Broker*. The *Inter-Cloud Exchange* delivers the request to the *Catalog* registering the service. An acknowledgement is returned to the *SP* advising the success or fail of the process.

4.1.2.4 Inter-Cloud Exchange or Market (Actor)

The *Inter-Cloud Exchange*'s aim is to manage the *Catalog* of the *Cloud Federation*. It is connected to the *Inter-Cloud Broker* and receives the request for searching and registering services. It also has to select and bind appropriate services; finally it is in charge of maintaining the monitoring information of providers and resources. The main use cases for the *Inter-Cloud Exchange* are: *Search Service*, *Match Service*, *Bind Service*, *Unbind Service*, *Monitor Service*, and CRUD operations (Create, Read, Update, and Delete) over a *Service* from the *Catalog.* A description of the related use cases are:

UC 17. **Search Service**: A request for service arrives to the *Inter-Cloud Exchange*; then a search request is delivered to the *Catalog* looking only which *SPs* has that service (not considering constrains or QoS parameters). Then a set of services and *SPs* are returned to the *Inter-Cloud Exchange* where a matching process will be executed.

UC 18. **Match Service**: A set of *SPs* and services are delivered to the *Inter-Cloud Exchange*. The matching service will select based on several parameters the combination that better match the original request for service done by the *SP*.

UC 19. **Bind Service**: Once a service has been selected, the *Inter-cloud Exchange* updates the availability of the service in the *Catalog* and advices to the selected provider through the *Inter-Cloud Broker*.

UC 20. **Unbind Service:** A request for termination or cancelling arrives to the *Inter-Cloud Exchange* updating the *Catalog*. If the request is for termination, it informs the requestor; but if it is for cancelling, it informs to the Provider.

UC 21. **Monitor Service**: A request for monitoring arrives from a *SP*, the *Inter-Cloud Exchange* responds with information about the *SP* assigned to the service. If the *Inter-Cloud* works with a reputation system, the information returned could be for example the level of trust of the *SP* or even some kind of ranking. Reputation systems are beyond the analysis of this work.

UC 22. **CRUD Service**: Creation, Read, Update and Delete (CRUD) operations are done over the *Catalog*. They are basic functions to work with the services registers.

The Use Cases described before are showed in Figure 1.

Figure 1: Federated Inter-Cloud Use Case diagram

4.2 Federated Inter-Cloud Pattern

4.2.1 Intent

Federated *Inter-Cloud* systems allow many *Service Providers (SPs)* to share resources under demand. *Consumers* request resources from a specific *SP*, which may in turn request them to the *Federation*. The *Inter-Cloud* solves the problem of the resources exhaustion, or the lack of specific services that can affect a *SP*. The *SPs* form a federation in order to share capabilities.

4.2.2 Example

One of the sites of our clients (*a.k.a. Consumer*) is experiencing a significant increase in shopping traffic, but its *SP* is unable to provide a greater number of machines as all resources are being used by other clients. Our client lost sales opportunities which could get him out of business in the long term. Even in the best of the cases, we may lose a client.

4.2.3 Context

Service Providers (SPs) are connected to the Internet. They can be reached by each other.

4.2.4 Problem

Some clients may need a larger amount of resources unexpectedly, or they may need a specialized service. In either case, the *SP* may not be able to satisfy the request. How can an *SP* be prepared for these events?

The solution to this problem must resolve the following forces:

- Transparency – *Consumers* should not be concerned about the process and setup of agreements between *SPs* or another component of the *Inter-Cloud*. From their point of view, they should feel that they have an unlimited pool of resources.
- Capacity – *Consumers* normally have access to enough computational power, only occasionally we should fail to provide what they want.
- Scalability – *Consumers* should be able to increase from a few to many the amount of resources used.
- Degree of Services – *SPs* may want to offer different types of service quality, some better than others (e.g., premium and standard).
- Autonomy—*SP* should be able to provide services to *Consumers* independently of others.
- Heterogeneity—the *SP* should be able to provide a much larger variety of services.

4.2.5 Solution

An *Inter-Cloud* system can satisfy requests of the *Consumers* when a single *SP* cannot do it by itself by providing extra resources for the subscribed *SPs*. It also increases the diversity of them and improves the elasticity of every *SP*. The *Inter-Cloud* must ensure the compliance of *QoS*, monitoring and managing the operations between the *SPs*.

4.2.5.1 Structure

Service Providers (**SPs**) forms a federation in order to share capabilities (*Inter-Cloud*). It keeps a **Catalog** of the resources available in the *Inter-Cloud* to decide how to satisfy them. The *Inter-Cloud* could be composed by federated **SPs** or by an association through a broker (acting as front-end and single point of access for every **SP**). Most *Inter-Cloud* projects use a centric component where the requests of the **SPs** are satisfied. Compliance and agreements would be specified in a kind of *SLA*. Figure 2 shows a class diagram for the *Federated Inter-Cloud* pattern. A **SP** processes the requests

from *Consumers* through a **Portal**. The **SP** performs requests through the **Inter-Cloud Broker**. The **Inter-Cloud Broker** implemented internally or externally redirects the request to the **Inter-Cloud Exchange**; both are part of the **Cloud Federation** system. The **Inter-Cloud Exchange** consults its **Catalog** (a.k.a. *Information Repository*) looking for the best match for the request. The **Catalog** lists several kinds of **Services**. The **Inter-Cloud Exchange** is aware of the condition of the entire system through status information coming from **Inter-Cloud Brokers**. The **Catalog** is updated considering the information sent by the **Inter-Cloud Brokers**.

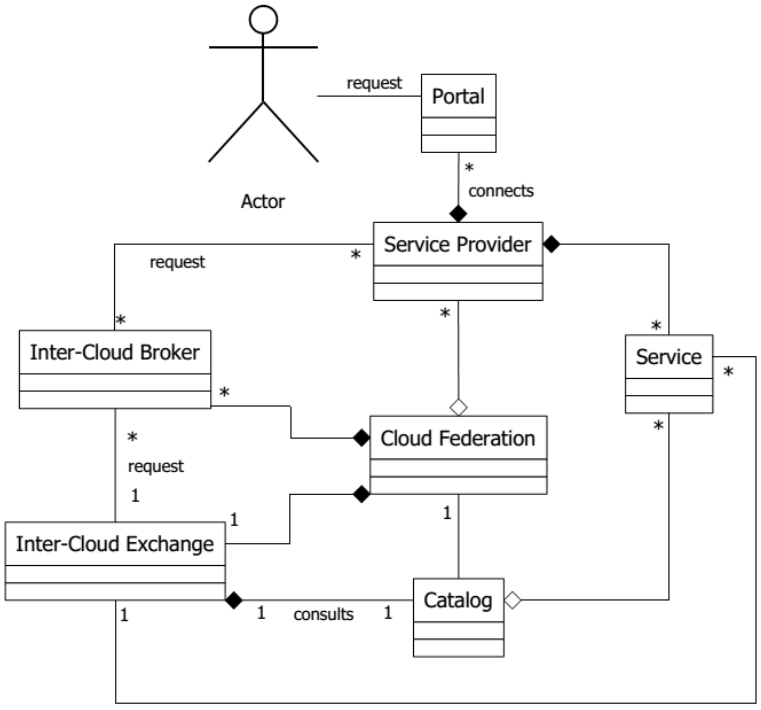


Figure 2: Federated Inter-Cloud components

4.2.5.2 Dynamics

Some Use Cases are the following:

- Request a resource/service (actor: Consumer)
- Join Federation (actor: Service Provider)
- Publish service (actor: Service Provider)
- Remove service (actor: Service Provider)
- Leave Federation (actor: Service Provider)

We show in detail one of these use cases below:

UC: Request a resource (Figure 3)

Summary: A *SaaS Consumer* requests to increment its amount of resources and its provider does not have enough capacity. The *Cloud Exchange* (a.k.a. *Market*) assigns the resources from one or many of the federation participants.

Actor*: Consumer*

Precondition: The *Service Provider* belongs to the federation. By agreement, a request is sends to the *Cloud Exchange* when no more resources can be assigned by the *Service Provider*.

Description:

a) A *Consumer* requests to increment its amount of resources.

b) The *Service Provider SP1* analyzes if it can provide the requested resources. If it cannot satisfy the *Consumer* request, a request is sent through the *Inter-Cloud Broker*.

c) The *Inter-Cloud Broker* redirects it to the *Inter-Cloud Exchange*.

d) The *Inter-Cloud Exchange* consults the *Catalog*.

e) The *Catalog* matches a suitable provider for the request and answers the request.

f) The *Cloud Exchange* communicates with the selected *Service Provider SP2 (*through the *Inter-Cloud Broker)*.

g) The *Service Provider SP2* communicates through the *Inter-Cloud Broker* the answer for its request to the *Service Provider SP1*.

h) The service is provided to the *Consumer*

Alternate flow: The request cannot be satisfied by the federation.

Post condition: The *Consumer* received the resources he requested.

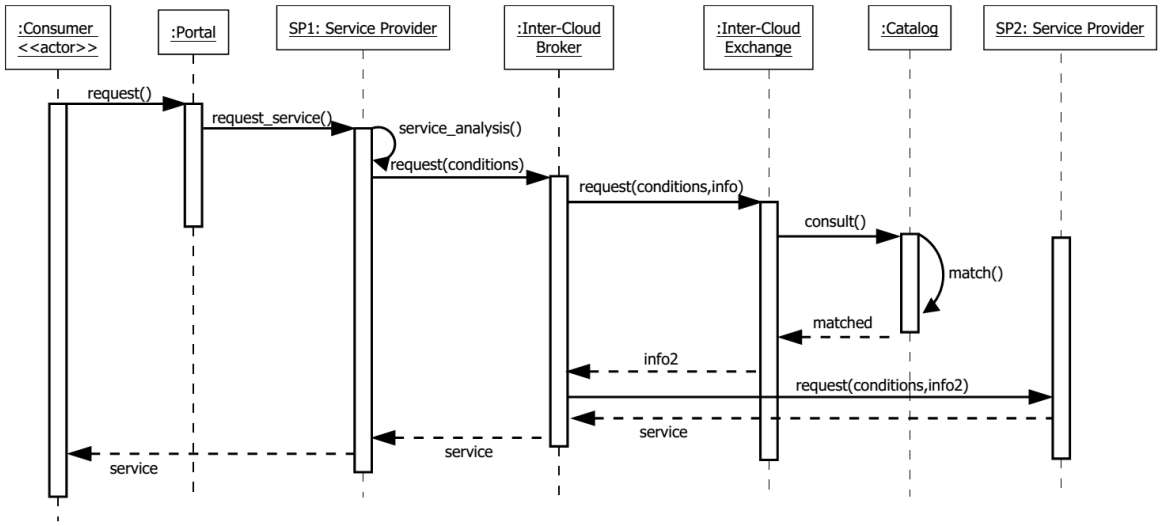


Figure 3: Consumer request a resource

4.2.6 Implementation

It is important to considerer that this pattern is appropriate when the amount of participants is low because of its centralized architecture (there is just one central component in charge of the entire federation). When the amount of *SPs* increases very much (might be reaching the thousands) a decentralized approach would be better option.

- An agreement in the form of a *Service Level Agreement* (*SLA*) is recommended in order to monitor, describe and assign the resources and be able to publish and match offerings. A *SLA* is a part of a service contract where a service is formally defined. *SLA* issues go beyond the scope of this work and will not be further discussed.
- The use of a *Cache* that stores the last/or most popular *virtual appliances* requested could improve the instantiation of resources when *IaaS* requests are made (improving the time of searching and matching). The *Federated Cloud Management* approach in [Kec12a] uses caches inside and outside of the *SP*. In [Kec12a] an external component captures the events from different *Cloud Brokers* in order to improve the assignment of resources.
- It is recommended to implement the *Catalog* and *Inter-Cloud Exchange* in the same component in order to reduce the network traffic and accelerate the update and matching process.
- The *Inter-Cloud Broker* could be shared or each *SP* might have its own *Inter-Cloud Broker*.

- It is good to have a variety of *SPs* (in the sense of the kind of services they offer) to satisfy consumers with special needs.
- The *SP* must adopt the *API* used by the *Inter-Cloud Broker* in order to be able to communicate.

4.2.7 Consequences

The *Federated Inter-Cloud pattern* provides the following benefits:

- Transparency – A *Consumer* never realizes he is dealing with a *Federation*. The *Inter-Cloud Broker* takes care of the negotiation process with the *Inter-Cloud Exchange* and of all other internal actions.
- Capacity- since the *SP* belongs to a federation of *SPs*, the amount of resources offered is increased.
- Scalability – We can accommodate much more *Consumers* and more requests for computational power than in the case of a single *SP*.
- Degree of Services- since the *SP* belongs to a federation of *SPs*, the kind of resources offered is increased, and even they can be combined to form different and exclusive services.
- Accessibility- Since the *Consumer* access is transparent to its *Cloud Computing* system, and considering that the last one has a single point of access to the *Federation* (*Inter-Cloud Broker*), it can also have access to its information from anywhere, even if spread it in different *SPs*.
- Autonomy - Each *SP* can service C*onsumers* independently of the others.
- Heterogeneity— if selected carefully, the *SPs* are able to provide a much larger variety of services than a single *SP*.

The *Federated Inter-Cloud pattern* has the following liabilities:

- The federation must constantly inform the *Catalog* of its available resources so they can be considered for assignment. This constant feeds from several components might increase the network traffic reducing the overall performance.
- Compliance - who is responsible of compliance with regulations? Monitoring systems must be implemented in order to guarantee the proposed services.
- Logging an auditing - Logging and auditing systems must be implemented in order to record and be able to reconstruct who accessed what and when. Who keeps logs? Systems must trust in a common component that takes care of the logging process.

- Different security models – Different *SPs* might have different security models. Some could implement for example, a role-based access control, while others may use an access matrix model for authorization. This could be solved by implementing a mapping function in the broker.
- SLAs - the *SP* must have *SLAs* with its providers (a cascade of *SLAs*). This is complex to manage.

4.2.8 Example Resolved

We are now able to handle peaks in the amount of resources needed by a client, even if the primary *SP* does not have enough amounts of resources. We can provide our customers with as many machines as the federation can provide us. The whole process is performed transparently to the client. We would be able to handle a larger number of clients.

4.2.9 Known Uses

- The *InterCloud* project [Buy10] was developed at the University of Melbourne. It is one of the first approaches to an *Inter-Cloud* system. It proposes a centralized Market around some *Cloud Coordinators* and *Cloud Brokers* which receive and redirect the requests and return responses from and to the *Market*. An *SLA* specifies the details of the service to be provided in terms of a metric agreed upon by all parties, and incentives and penalties for meeting and violating the agreements, respectively.
- The *Contrail* project [Car12] proposes a complete framework for cloud federations focusing on authentication and authorization. This project was supported by the *FP7* project's Open Computing Infrastructures for Elastic Services. This work is one of the most complete *Inter-cloud* proposals. It uses a component called *Federation Runtime Manager* as a single front end for all services, dedicated to map user requests to cloud resources. This approach uses multiple *Federations Access Points* distributed over the network, each one potentially in contact with all SPs in the federation. This project also offers the possibility (in contrast to the others) to federate or not resources using specials adapters in the *Federation Access Points*, so this project works forming *Federated Inter-Clouds* and *Multiclouds*.
- *Dynamic Cloud Collaboration Platform* [Meh10] is a project supported by the *Future-based Technology Development Program*. This approach focus on heterogeneous and dynamics aspects of the *Cloud Computing* environment. In this system the consumer interacts transparently with the *Virtual Organization based dynamic collaboration platform* (*VO-based* component). This system includes a

Catalog inside the *SP* which helps to answer the requests coming directly from the consumer to their subscribed SP. The *SP* are federated forming a dynamic collaboration network for sharing resources using the *VO-based* component to resolve and assign the best combination of *SPs*.

- The *Federated Cloud Management* project [Kec12a] proposed a single point of access to the federation called *Generic Meta Broker Service*. This component takes care of the submission of a request from the *Consumers* and worries about matching it with the best possible provider. The *Federated Cloud Management* uses an external component, a database (the *Federated Cloud Management Repository*) and internals caches (*Native repository)* which improve the performance of the system by avoiding virtual images to be constantly moving in the system (because the cache saves them for possible reuse). The *Generic Meta Broker Service* uses external *Cloud Brokers* for instantiation and to redirect the calls to the *SPs*. In [Kec12b] an extra component is included to improve the way the *Generic Meta Broker Service* makes decisions and matching (the *Global Autonomous Manager*), by monitoring and analyzing the information arrived from the *Cloud Brokers*.

4.2.10 Related Patterns

- The Circle of Trust pattern [Del07] allows the formation of trust relationships among *SPs* to allow their subjects to access an integrated and more secure environment.
- The Identity Federation pattern [Del07] allows the dynamic creation of identities within an identity federation consisting of several *SPs*. Therefore, identity and security information about a subject can be transmitted in a transparent way for the consumer to *SPs* from different security domains.
- The Secure Broker pattern [Fer13a], extends the Broker pattern to provide secure interactions between distributed components.

4.3 Inter-Cloud Broker pattern

4.3.1 Intent

The *Inter-Cloud Broker* pattern can be used to structure an *Inter-Cloud* system of multiple and heterogeneous *Service Providers* (*SPs)*. An *Inter-Cloud Broker* component is responsible for enabling and managing the communication between *SPs* such as forwarding requests, easing the registration of services and for transmitting results and exceptions.

4.3.2 Example

A set of *SPs* agree on sharing resources because this would improve the amount and variety of services they offer to their clients. They want to create a federation of resources and to be able to collaborate and interoperate. However, they are totally heterogeneous and cannot modify the protocols they use because their applications are tightly coupled to them.

4.3.3 Context

In an *Inter-Cloud* system, resources are shared by multiple heterogeneous *SPs* which belong to a Federation and autonomously cooperate with each other. The requested resource can be at the *SaaS, PaaS,* or *IaaS* level. The *Inter-cloud Broker* is located between *SPs* and in contact with the *Inter-Cloud Exchange*. The *Inter-cloud Exchange* is where each resource is registered in a catalog, and where every request and resource assignment are bound (matching process).

4.3.4 Problem

A set of *SPs* agree in sharing resources, but they are distributed and heterogeneous; they cannot agree on a common protocol to communicate and interoperate with each other because their services are tightly coupled to them. Also, neither wants to limit the amount of *SPs* that could join to the federation. How to allow the communication and interoperation of multiple and heterogeneous *SPs* in a scalable way? *SPs* should not have to worry about the contracts and technical features of the federation and the way they communicate each other.

The solution is affected by the following forces:

- Transparency – *SPs* should not have to worry about a specific kind of protocol for communicating with others.
- Remote – *SPs* should be able to access services provided by others through remote service invocations.
- Location-independency – *SPs* should be able to access services independently of their location, even if the service changes its location.
- Dynamic operation – *SPs* could be added or removed at run-time.
- Scalable – The system should be able to scale.
- Management – The system should be able to be easily configured, monitored, and maintained.
- Fault-Tolerance – The system should be able to keep working if a single component fails.

4.3.5 Solution

Use an intermediate component that allow the intercommunication between distributed components. The component make the services available to others using common interfaces or protocols, transmitting results and exceptions back to the requestor. *SPs* register themselves through this component (which coordinates the registration of services using the *Inter-Cloud Exchange*, that is also in charge of match and establish relations between *SPs*).

4.3.5.1 Structure

The *Inter-Cloud Broker* pattern is composed by six types of participating components: **Service Providers** (they can be *Clients* and *Servers*), **Inter-Cloud Broker**, **Inter-Cloud Exchange**, **Client-side Proxy**, **Server-side Proxy** and **Bridge**. The **Inter-Cloud Broker** is located between **Service Providers** and also is connected to an **Inter-Cloud Exchange** component in order to allow the communication and negotiation between them. The **Inter-Cloud Broker** redirects the requests coming from **Service Providers** to the **Inter-Cloud Exchange** component. The **Inter-Cloud Exchange** is aware of the conditions of the entire system through status information coming from the **Inter-Cloud Broker**.

The **Bridge** component, could be added to improve the interoperation, fault-tolerance and decentralization of the current solution by connecting several **Inter-Cloud Brokers**.

Figure 4 show a class diagram of an **Inter-Cloud Broker**.

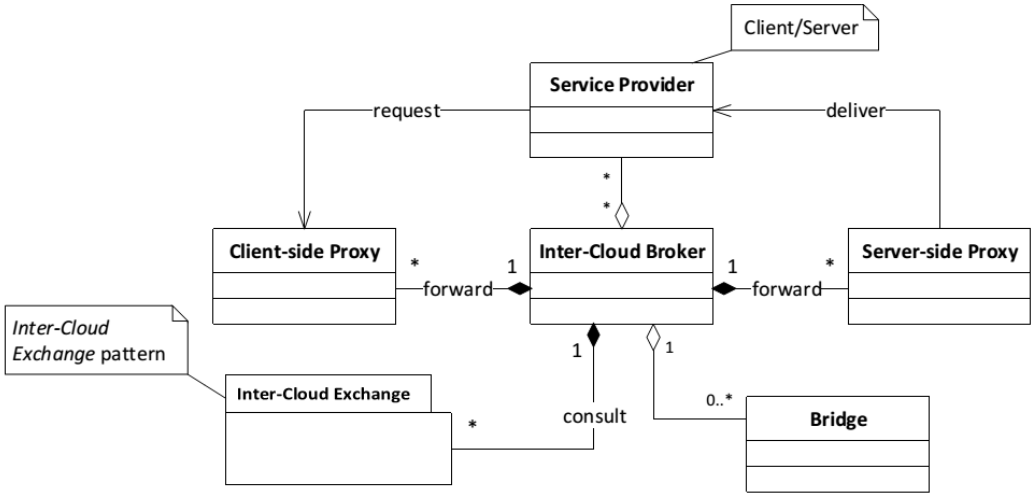


Figure 4: Class diagram of an Inter-cloud Broker

Service Providers could be *Clients* or *Servers* as well (even both at the same time). *Clients* are applications that request access to the services of at least one *Server*. After the service has been

provided they receive responses or exceptions through the **Inter-Cloud Broker**. Due to the indirection of the solution (a broker that intermediate the communication), *Clients* do not need to know the location of *Servers* or services they access, allowing easily the addition of new services and the movement of existing services to other locations, even while the system is running. The **Inter-Cloud Broker** pattern offers an *API* to its **Service Providers** that include operations for registering services and mechanisms for finding and invoking *Server* resources. Responses and exceptions from a service execution are forwarded by the **Inter-Cloud Broker** to the **Service Provider** that sent the request, unless the *Inter-Cloud* allow the direct communication of the response once the initial communication is established.

Client-side Proxies represent a layer between *Clients* and the **Inter-Cloud Broker**. This additional layer provides transparency to the requestor, making believe to the *Client* that the service is provided as a local one, reducing load and scaling better. **Client-side Proxy** or **Service-side Proxy** allow the hiding of implementation details from the *Clients* such as the inter-process communication mechanism used for message transfer between *Clients* and the **Inter-Cloud Brokers** and the formatting and packing of parameters and results. **Server-side Proxy** is analogous to the **Client-side proxy**. The difference is that they are responsible for receiving requests, unpacking incoming messages, translate parameters, and calling the appropriate service. They are used in addition for formatting and packing results and exceptions before sending them to the *Client*.

When results or exceptions are returned from a *Server*, the **Client-side proxy** receives the incoming message from the **Inter-Cloud Broker**, unpack the data and forward it to the *Client*. The response could be provided in two different ways, directly or indirectly. When direct communication is used, the response goes straightforward to the *Client*, otherwise, the result pass first through the **Inter-Cloud Broker** that forward it to the *Client* (indirectly). The direct communication option achieve better performance, but it is necessary that both, *Client* and *Server* use and understand the same protocol for interoperation.

4.3.5.2 Dynamics

We show here two fundamental use cases for the Inter-Cloud Broker pattern:

- *UC: Service Provider* registers a service (actor: Service Provider) (Figure 5)
  Summary: A **Service Provider** request to register a service.
  Actor**: Service Provider**
  Precondition: The **Service Provider** belongs in advance to the federation and has adopted the **Client-side Proxy** component.

Description:

a) The **Service Provider** creates a request for the registering of a service.

b) The **Service Provider** collects additional information like constrains and deliver the request to the **Client-side Proxy**.

c) The **Client-side Proxy** format and pack the information of the service to be registered and forward it to the **Inter-Cloud Broker**.

d) The **Inter-Cloud Broker** deliver the request to the **Inter-Cloud Exchange**.

e) The **Inter-Cloud Exchange** registers the service and acknowledges it to the **Inter-Cloud Broker**.

f) The **Inter-Cloud Broker** forward the response to the **Client-side Proxy** of the requestor.

g) The **Client-side Proxy** deliver the acknowledgement to the **Service Provider**.

Post condition: The **Service Provider** register the service in the **Inter-Cloud Cloud Exchange** and now is available for others to be found and use.

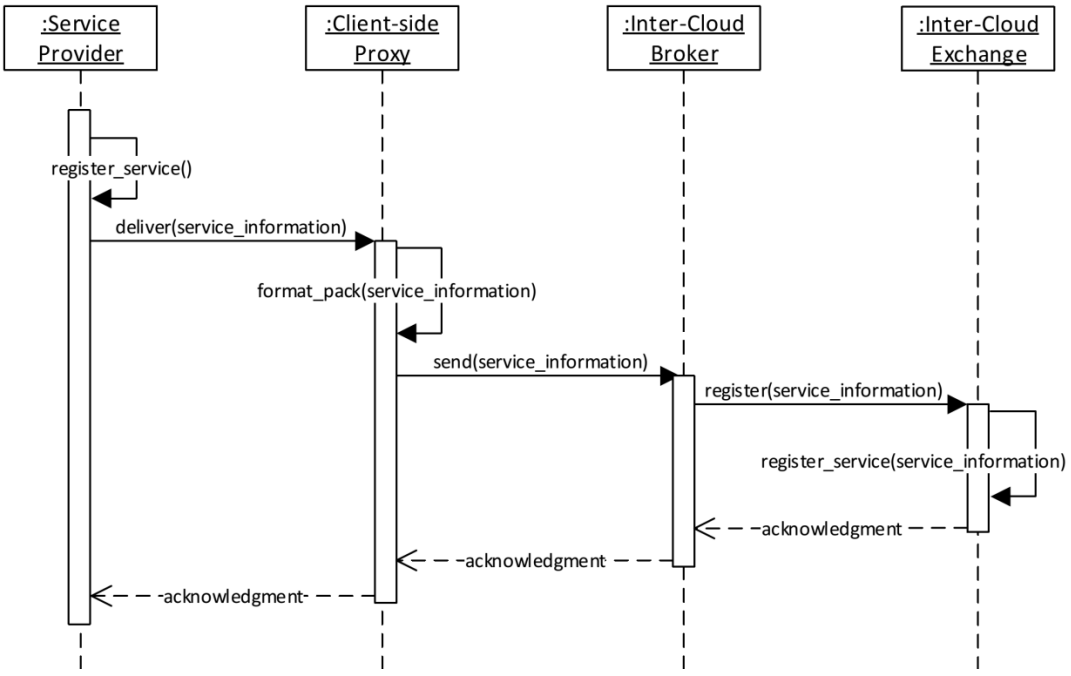


Figure 5: The Service Provider registers a service (Inter-Cloud Broker)

- *UC:* Request a Service to the Inter-Cloud (actor: Service Provider) (Figure 6)

Summary: A **Service Provider** request a service to the Inter-Cloud.

Actor**: Service Provider**

Precondition: The service requested has been previously registered. The conditions, requirements and constrains of the service have already been specified. Exists a **service Provider B** that can deliver the requested service and that also satisfies the requirements of the **service Provider A**.

Description:

a) The **service Provider A** requests a service to the Inter-Cloud delivering the request to its **Client-side Proxy**.

b) The **Client-side Proxy** formats and packs the information of the service to be requested forwarding it to the **Inter-Cloud Broker**.

c) The **Inter-Cloud Broker** delivers the request for service to the **Inter-Cloud Exchange**.

d) The **Inter-Cloud Exchange** search, match and notify to the **Inter-Cloud Broker** the location of the service requested.

e) The **Inter-Cloud Broker** notify the **Server-Side Proxy** of the target provider, who unpacks the message and delivers it to its **service Provider B.**

f) The service is transmitted through the **Server-side Proxy** directly to the **Client-side Proxy** of the **service Provider A.**

g) The **Client-side Proxy** unpacks the information and deliver the service to the **service Provider A.**

Alternate Flow: The service requested is delivered indirectly to the **Service Provider.**

Post condition: The service requested is delivered directly to the **Service Provider**.

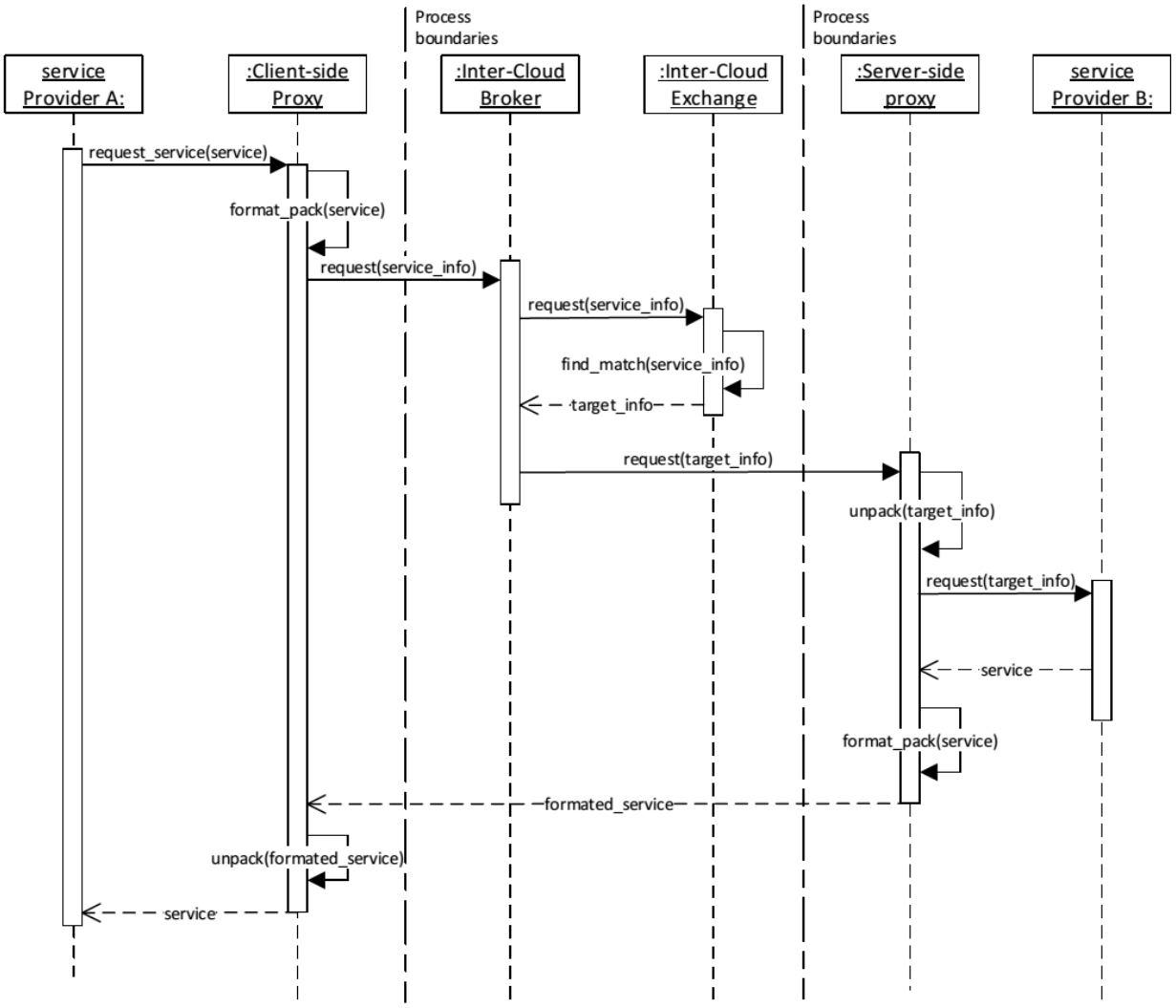


Figure 6: A Service Provider request a service

4.3.6 Implementation

- The *Client-Side Proxy* and *Server-Side Proxy* could be implemented internally [Buy10, SIIF] or externally [Car12, Kec12a] to the *SP*. When is implemented externally, the *Client* and *Server-Side Proxies* usually appears together in one unique component in a frontend fashion, concentrating all the requests and also delivering the services to the appropriate requestor (indirectly) [Car12, Kec12a].
- It is possible to improve the fault tolerance by replicating the *Inter-Cloud Broker* and *Inter-cloud Exchange*, establishing a communication method between them [SIIF].
- Collaboration between *Inter-cloud Brokers* and between *Inter-cloud Exchanges* could be achieved using *Bridge* components for each [SIIF].

- The decoupling between the *Inter-cloud Broker* and the *Inter-cloud Exchange* provides better fault tolerance to the system, also, helps in the load balancing between them, managing information of different and independent administration zones [SIIF].
- When *Inter-cloud Brokers* work in different administrative zones, it is possible to establish a kind of cache memory that stores local information about the *SPs* under its domain, even it could manage a local catalog with the services offered, decreasing the time and traffic generated for repetitive requests [SIIF]. In this scenario is necessary the constant updating of information between *Inter-cloud Broker* and the *Inter-cloud Exchange*.
- In [Car12] are used *Internal* and *External Adapters* (equivalent to the *Client* and *Server-side Proxy* in this work) but only *Internal Adapters* allow the sharing of resources (federated inter-cloud) while *External Adapters* are for *Multi-Cloud* purposes. Hybrid solutions are also possible when a *SP* don't want to share resources but want to keep using external services.

4.3.7 Consequences

The *Inter-Cloud Broker pattern* provides the following benefits:

- Location Independence – *SPs* does not have to worry about where a service is located or even if that service is moved. Equally, *SPs* do not have to know the location of their clients, they receive every request from the *Inter-Cloud Broker* associated to them.
- Changeability and Extensibility components – Does not matter if the *SPs* changes the protocols they use as long as the *Proxies* of the system understand it. *The Inter-Cloud Broker* API does not have to be changed.
- Interoperability and Integration between *Inter-Cloud Brokers* systems – As long they use the same API different *Inter-Cloud Brokers* systems could be integrated between them improving the amount and variety of resources offered.
- Fault Tolerance – Due to the fact that the *Inter-Cloud Exchange* is decoupled of the *Inter-Cloud Broker* and also the replication of components and information is allowed (because of the use of the *Bridge* component), is possible to have an improvement in the reliability when comparing with the classic *Broker Pattern*.
- Scalable – Due to the replication of information and the communication between the *Inter-Cloud Exchange*, *Bridges*, and the decoupling of the *Inter-Cloud* Exchange and

Inter-Cloud Broker, the *Inter-Cloud* now can scale according to the incoming clients, decentralizing the *Inter-Cloud* operations.

- Hide details – The architecture hides system implementation details due to the use of the *Client* and *Server-side Proxys* components that translates and operates all the communications with the external environment.
- Dynamic – Once the service has been completed or before to initialize it, is possible to add, update or remove the service from the federation. This is possible with the *Inter-Cloud Exchange* that records the location, existence and availability of every service offered.
- Remote – With the use of the *Client* and *Server-side Proxys* all operations related with the *Inter-Cloud* are presented as local to all *SPs* (clients and servers), but they are really distributed over the network.
- Management – Due to all components share the same API for communication, is possible to append a metering system for monitoring and also establish an upgrade system through replication, automating the update process.

The *Inter-Cloud Broker pattern* provides the following liabilities:

- Efficiency- Any system that uses the *Inter-Cloud Broker* pattern will suffer of a bounded efficiency due to the use of the multiple indirections and the overhead product of the multiple formatting and packing operations. A system that is flexible and portable to multiple interfaces and protocols is usually less efficient than one static and known.

4.3.8 Known Uses

- In the *IEEE Draft document for Inter-Clouds systems* [SIIF] the brokering process is implemented using three components. An *InterCloud Gateway* that belongs to each *SP*, an *InterCloud Exchange* and an *Intercloud Root*. *InterCloud Gateway*s communicates directly to each other, they are in charge of the translation between the heterogeneous *SP*. They must understand each other using the same protocols (like the *Client* and *Server-side Proxy* in this work). The *InterCloud Gateway*s are also connected to the *InterCloud Root*, this one holds also the *InterCloud Exchange* component inside of him. The *InterCloud Root* is the equivalent to the *Inter-Cloud Exchange* component exposed in this pattern, it has to manage the resource directory services (that will be, the trust authority service and the presence information).

- The *InterCloud* project [Buy10] proposes three components as part of the brokering process. A *Cloud Coordinator* component inside of a *Cloud Computing* environment or any IT enterprise that want to publish offers about the resources available for sharing. The user asks to its *Cloud Broker* component its intentions and this one communicates and negotiate with the *Cloud Exchange* the terms of the service to be provided, later, the *Cloud Coordinator* communicates directly to the *Cloud Broker* that start the delivering of the requested service.
- The *Contrail* project [Car12] uses a component called *Federation Runtime Manager* as a single front end for all services, dedicated to map user requests to cloud resources. This approach uses multiple *Federations Access Points* distributed over the network, each one potentially in contact with all *SPs* in the federation. The *Access Points* are connected to each other (acting like the *Bridges* exposes in this pattern) in order to replicate the information between them.
- In the *Dynamic Cloud Collaboration Platform* [Meh10] the brokering process is achieved using two components, the *primary Cloud Provider* (the initiator of the *Inter-Cloud* and the one that holds the *Catalog*) and the *Virtual Organization based dynamic collaboration platform* (*VO-based*) that is in charge of deliver the services to the requestors. The request for services are strictly directed to the *primary Cloud Provider* that is in contact with the others *Cloud Providers* that belongs to the federation, this one will decide the best combination of services to users. Later, the service will be provided through the *VO-based* component to the requestor.
- The *Federated Cloud Management* project [Kec12a] uses three components to achieve the communication between *SPs*. The *Generic Meta Broker Service* acts like the *Cloud Exchange*, responsible of selecting a suitable provider for the requested service. The *Generic Meta Broker Service* communicates to a specific *Cloud Broker* the information about the request done. The *Cloud Broker* acts like the *Client* and *Server-side Proxy* of each *SP*, it is used for the submission and delivering of services communicating directly with its *SP*. For each *SP* the *Cloud Broker* is setup externally and talks with a *Handler* implemented internally that will process the request inside the *Provider*.

4.3.9 Example Resolved

We have now a way to communicate between every heterogeneous *SP* independently of the protocol they use. The *Inter-Cloud* now can adopt a large variety of *SPs* and offer more and diverse services to its clients in a scalable way.

4.3.10 Related Patterns

- *Federated Inter-Cloud pattern* [Enc14a] – solves the problem of resource exhaustion, or the lack of specific services that can affect at one *SP*. The *SPs* form a federation in order to share capabilities.
- *Broker pattern* [Bus96] – solves the problem of having in a distributed environment a set of heterogeneous and independent components that needs cooperation between them.
- *Broker Revisited* [Kir04] – review of the *Broker Pattern* that extract the registration of services from the *Broker pattern* connecting clients with remote objects using invocations from clients. It suggest the use of the *Lookup Pattern* to achieve the registration of services.
- *Secure Broker* [Fer13a] – solves the problem of how to provide secure interactions between distributed objects. This pattern is an extension of the *Broker pattern*.
- *Lookup Pattern* [Kir04] – solves the problem of how to find and retrieve initial references of distributed objects or services. This pattern was used in the composition of the *Inter-cloud Exchange pattern* [Enc14e].

4.4 Inter-cloud Exchange pattern

4.4.1 Intent

The *Inter-Cloud Exchange* pattern can be used to structure a federated *Inter-Cloud*. An *Inter-Cloud Exchange* is responsible for enabling and managing the registration of *SPs* and their services, as well as the lookup and matching of specific requests for services.

4.4.2 Example

A set of *Service Providers (SPs*) agree on sharing resources. The amount of SPs and services in the federation could increase too much, and they cannot be responsible for managing and coordinating the entire catalog of services and neither managing the matching of services due to the extensive list of items and the dynamic environment.

4.4.3 Context

In an Inter-Cloud system, resources are shared by multiple heterogeneous SPs which belong to a Federation and autonomously cooperate with each other. The requested resource can be at the SaaS, PaaS, or IaaS level. The *Inter-Cloud Exchange* component is directly in contact to the *Inter-Cloud Broker* component. The *Inter-Cloud Broker* pattern enables the communication between distributed and heterogeneous *SPs* and the communication with the *Inter-Cloud Exchange*.

4.4.4 Problem

A set of *SPs* agree on sharing resources, but due to the possibly high amount of participants and services, and because of the dynamic interaction between them, they cannot manage or coordinate individually all the information about the resources offered and their status, neither perform matching between services and providers. How can we allow the registration, lookup and matching of services without copying the whole catalog of resources in every provider, simplifying administration and communication in a scalable way?

The solution is affected by the following forces:

- Availability – a *SP* should be able to find out on demand what services are available in the federation.
- Transparency – *SPs* should not have to worry about adopting a specific kind of protocol for registering, searching or matching services.
- Remote – *SPs* should be able to register, lookup and match services provided by others through remote service invocations.
- Simplicity- the solution should not burden a *SP* obtaining locations of services nor a server providing the references.
- Location-independency – *SPs* should be able to register, lookup and match services independently of their location; even if the service changes its location.
- Dynamic – *SPs* and their services could be added or removed at any time and their presence should be reflected by others *SPs* in run-time.
- Scalability – *SP* and services should be able to increase from a few to many.

4.4.5 Solution

The solution is to use a specialized, remote and independent component for registering *SPs* and services, finding and matching services between requests and offerings.

4.4.5.1 Structure

The *Inter-Cloud Exchange* pattern is composed by three types of participating components: **Service Providers**, an **Inter-Cloud Broker** and one or more **Inter-Cloud Exchanges**. Each **Inter-Cloud Exchange** is connected directly to one **Inter-Cloud Broker**. The **Inter-Cloud Broker** is the channel that allows the communication and negotiation between **Service Providers** and the **Inter-Cloud Exchange**. The **Inter-Cloud Broker** redirects the requests coming from **Service Providers** to the **Inter-Cloud Exchange**. A common interface is set between the **Inter-cloud Exchange** and the **Inter-cloud Broker** to allow the transmission of results, registrations and exceptions to the corresponding *SPs.* The **Inter-Cloud Exchange** is aware of the conditions of the entire system through status information coming from the **Service Providers**. Figure 7 show a class diagram of an **Inter-Cloud Exchange**. The **Catalog** is a database that records the information about the *SP*s and the services associated to them, it also allows the lookup operations coming from the **Inter-cloud Exchange**. **Service Providers** could be *Clients* or *Servers* (even both at the same time). *Clients* are applications that request access to the services of at least one *Server*.

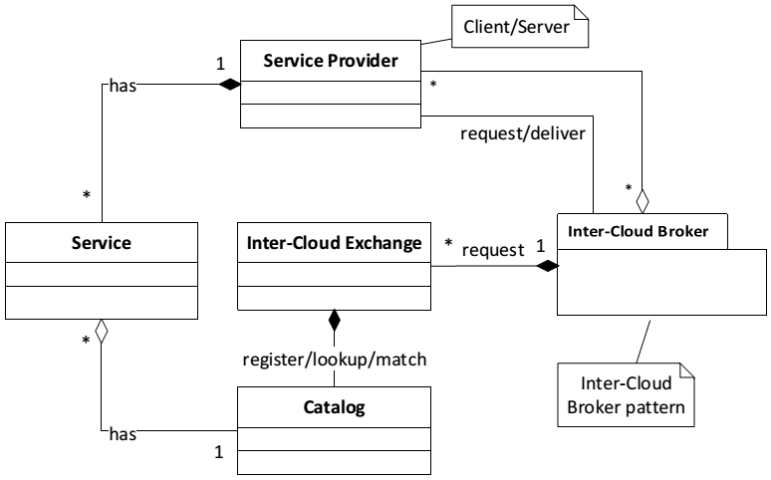


Figure 7: Class diagram of an Inter-cloud Exchange

4.4.5.2 Dynamics

We show here two fundamental use cases for the *Inter-Cloud Exchange* pattern:

- *UC:* **Service Provider** registers a **Service** (actor: **Service Provider**) (Figure 8)
  Summary: A **Service Provider** request to register a service.
  Actor**: Service Provider**
  Precondition: The **Service Provider** belongs in advance to the federation and has adopted the **Inter-Cloud Broker** pattern.
  Description:

a) The **Service Provider** creates an instance of **Service**.

b) The **Service Provider** request to register a service through the **Inter-Cloud Broker**.

c) The **Inter-Cloud Broker** forward the request for registration to the **Inter-Cloud Exchange**.

d) The **Inter-Cloud Exchange** executes a *registration request* to the **Catalog** delivering the needed information. The registration is made in the **Catalog** and an acknowledge message is sent to the **Inter-Cloud Exchange**.

e) The **Inter-Cloud Exchange** acknowledge the registration of the **Service** to the **Inter-Cloud Broker**.

f) The **Inter-Cloud Broker** forward the response to the **Service Provider**.

Post condition: The **Service Provider** register the service in the **Inter-Cloud Exchange** and now it is available for others **Service Providers** in the federation.

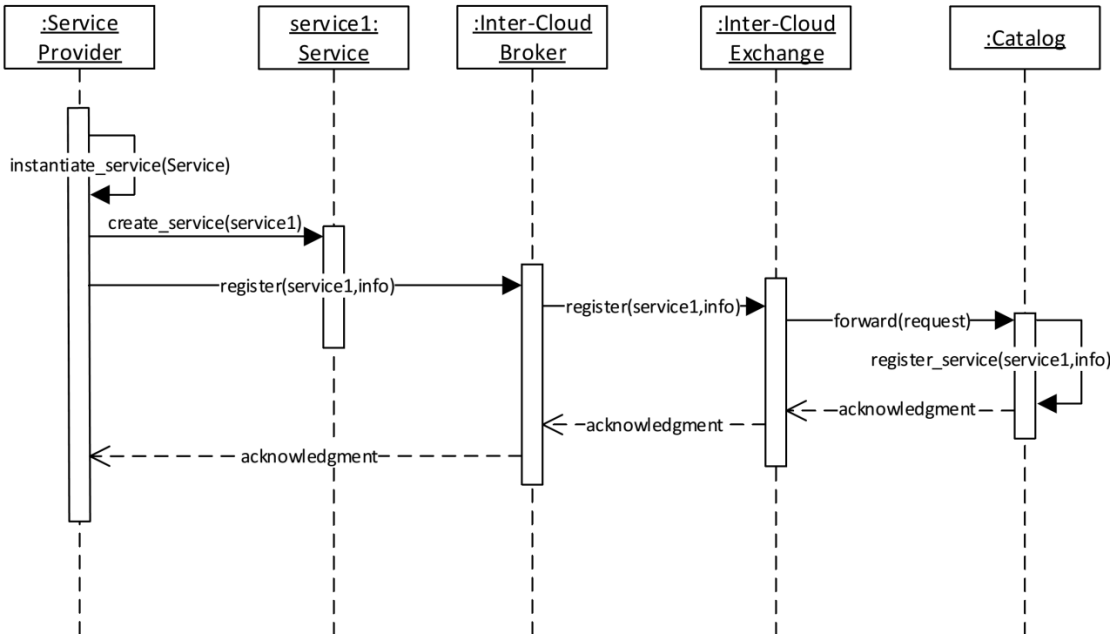


Figure 8: The Service Provider registers a service (Inter-Cloud Exchange)

- *UC:* Lookup for service (actor: Inter-cloud Broker) (Figure 9)

  Summary: An **Inter-cloud Broker** performs a lookup for service in the **Inter-cloud Exchange.**

  Actor**: Inter-cloud Broker**

  Precondition: **Service Provider** has requested a service to the **Inter-cloud Broker**. Conditions, requirements or constrains are provided by the **Service Provider** to the

Inter-cloud Broker. The desired service has been previously registered in the **Inter-Cloud Exchange** by another provider.

Description:

a) The **SP1** request a service through the **Inter-Cloud Broker**.

b) The **Inter-Cloud Broker** lookup for a service in the **Inter-Cloud Exchange**.

c) The **Inter-Cloud Exchange** forwards the search to the **Catalog**.

d) The **Catalog** searches candidate services in its repository and returns it to the **Inter-Cloud Exchange**.

e) The **Inter-Cloud Exchange** performs a matching process over the list of candidates provided by the **Catalog**, selecting one of them.

f) The **Inter-Cloud Exchange** delivers the information of the selected service to the **Inter-Cloud Broker**.

Post condition: The service information about the service requested by the **Inter-Cloud Broker** is delivered to him.

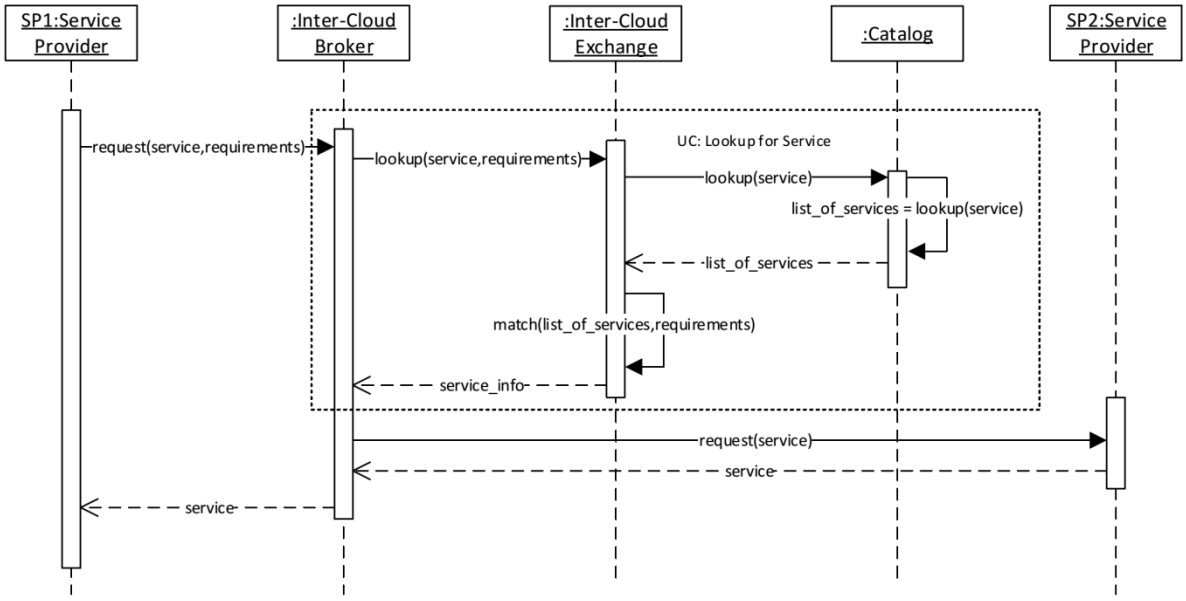


Figure 9: Inter-Cloud Broker lookup for a service

4.4.6 Implementation

- The **Inter-Cloud Broker** directly contact the **Inter-Cloud Exchange** to retrieve or forward the request made. The **Inter-Cloud Exchange** saves or updates all the information about the location and availability of the services offered.
- The **Inter-Cloud Broker** shares the *API* with the **Inter-Cloud Exchange** (and to the **Service Providers** as well), that include operations for registering services and

mechanisms for finding *Server* resources. Responses and exceptions are forwarded by the **Inter-Cloud Broker** to the **Service Provider** that sent the request.

- Because *Inter-Cloud* systems are meant to be working in a business/Market fashion, it is necessary to support every agreement with some document or contract that makes explicit every need and compromise. *Service Level Agreement* (*SLA*) is the most used way to define the contractual relationships and agreements between two parties, the provider and the client [Buy10, Cal10, Dem12 and Gro12]. An improvement in the *Inter-Cloud Exchange* would be related to the storage of *QoS (Quality of Service)* metrics promised by the providers, those metrics would be used to perform the matchmaking process when a request for service arrives.
- The matchmaking process (find the best combination between the offerings) could be done using heuristics algorithms, because in some cases the search space is too large for a complete revision [Sot11].
- In order to facilitate the matchmaking process, the *Inter-Cloud Exchange* could work with a *Distributed Hash Table* (*DHT*), maintaining distributed the information between multiple *Inter-Cloud Exchanges* nodes [SIIF], this would facilitate and optimize the matchmaking process.
- It is possible to improve the fault tolerance by replicating the *Inter-Cloud Exchange* establishing a communication method between them [SIIF].
- Collaboration between *Inter-Cloud Brokers* and between *Inter-Cloud Exchanges* could be achieved using *Bridge* components [SIIF]. The *Bridge* has been described in the *Inter-Cloud Broker* pattern.
- The decoupling between the *Inter-Cloud Broker* and the *Inter-Cloud Exchange* provides better fault tolerance to the system, also, helps in the load balancing between them, managing information of different and independent administration zones [SIIF].
- When *Inter-Cloud Brokers* work in different administrative zones, it is possible to establish a kind of cache memory that stores local information about the *SPs* under its domain, even it could manage a local catalog with the services offered (in local *Inter-Cloud Exchange* components), decreasing the time and traffic generated for repetitive requests [SIIF]. In this scenario it is necessary the continuously updating of information between *Inter-Cloud Broker* and the *Inter-Cloud Exchange*.

4.4.7 Consequences

The *Inter-cloud Exchange pattern* provides the following benefits:

- Availability – *SPs* access under demand to the *Inter-Cloud Exchange* through the *Inter-Cloud Broker* when a resource is needed and a request for service is launched.
- Transparency – Since the *Inter-Cloud Exchange* pattern belongs to the *Inter-Cloud Broker* pattern, every *SP* in the federation could perform registering and searching of services using the same interface used with the *Inter-Cloud Broker*. The matching process is done automatically in the *Inter-Cloud Exchange*.
- Remote – The *Inter-Cloud Exchange* is located outside *SPs*, and every request made is done through the *Inter-Cloud Broker*. The *Inter-Cloud Broker* is in charge of forwarding the request about the registering and searching of services (the match process is done automatically when a search is performed).
- Simplicity – The solution did not seeks initial references nor a server that has it (*SPs* does not have to download a list with a catalog of resources, for example), every request is managed by the *Inter-Cloud Broker* that forwards it to the *Inter-Cloud Exchange*.
- Location Independence – Since *SPs* use the *Inter-cloud Broker* for registering, searching and matching services, they do not need to know the location of *SPs* or services, allowing easily the addition of new services and the migration of existing services to other locations, even while the system is running. The *Inter-Cloud Broker* manages the location of the *Inter-cloud Exchange.*
- Scalable – Due to the replication of information and load balancing, the *Inter-cloud Exchange* can easily scale decentralizing its information, clients and operations. The *Inter-cloud Exchange* use the *Bridge* component (of the *Inter-cloud Broker*) to achieve these tasks.
- Dynamic – Even when the system it is running is possible to add, update, relocate or remove *SPs* or services from the catalog. This is possible just by invoking requests to the *Inter-Cloud Exchange* that records the location, existence and availability of every service offered.

The *Inter-Cloud Exchange pattern* has the following liabilities:

- Performance- The performance of this solution is related mostly with the fault tolerance in a centralized architecture (one single point of failure, but tackled with replication schemes), the searching time when the amount of *SPs* and services grows

too much and the inherent delay in getting a response because the information is in a remote site.

4.4.8 Known Uses

- In the *IEEE Draft document for Inter-Clouds systems* [SIIF], the *InterCloud Root* is the equivalent to the *Inter-Cloud Exchange* component exposed in this document, it has to manage the resource directory services (that will be the trust authority service and the presence information). The *InterCloud Exchange* has to make also the matching of resources applying preferences and constrains. This component also has to keep the Trust Domain information between the *SPs* affiliates.
- The InterCloud project [Buy10] use *Cloud Brokers* for contact the *Cloud Exchange* or *Market* (equivalent to the *Inter-cloud* Exchange in this work) and delivers the terms of the service to be provided; then the *Cloud Exchange* will search and match a *SP* and negotiates directly with it the terms and conditions of the association. The *Cloud Exchange* also manages the billing process.
- The *Contrail* project [Car12] uses a component called *Federation Runtime Manager* as a single front end for all services, that is dedicated to map user requests to cloud resources (the same functions of the *Inter-cloud Exchange*). This approach uses multiple *Federations Access Points* distributed over the network, each one potentially in contact with all *SPs* in the federation.
- In the *Dynamic Cloud Collaboration Platform* [Meh10] the *primary Cloud Provider* acts like the *Inter-cloud Exchange*, holding the *Service Catalog* in the *Information Repository* component; also it manages the registering process with the *Service Registry*, the *Policy Repository*, a *Mediator*, the *Price Setting Controller*, and the *Admission and Bidding Controller* all inside the same component.
- The *Federated Cloud Management* project [Kec12a] uses a *Generic Meta Broker Service* that acts like the *Inter-cloud Exchange* exposed in this pattern, responsible for selecting a suitable provider of the requested service. The *Generic Meta Broker Service* communicates to a specific *Cloud Broker* the information about the request done. The *Generic Meta Broker Service* is also connected with the *Federated Cloud Management Repository* that holds the catalog, including dynamic and static information about the *SPs*.

4.4.9 Example Resolved

Now, *SPs* can delegate the managing and storage of resources to the Inter-cloud Exchange through the *Inter-Cloud Broker*. The *SPs* do not need any special API to communicate with the *Inter-Cloud Exchange*, because all negotiations are made by using the *Inter-Cloud Broker*. The information about resources and the matching process is now concentrated in one place.

4.4.10 Related Patterns

- *Federated Inter-cloud pattern* [Enc14a] – solves the problem of resource exhaustion, or the lack of specific services that can affect at one *SP*. The *SPs* form a federation in order to share capabilities.
- *Inter-cloud Broker pattern* [Enc14b] – solves the problem of having a set of heterogeneous and distributed set of *SPs* that wants communication between them and also with a repository with information about shared resources. This pattern wraps the *Inter-Cloud Exchange* pattern presented in this work.
- *Lookup Pattern* [Kir04] – solves the problem of how to find and retrieve initial references of distributed objects or services. This pattern was used in the composition of the *Inter-Cloud Exchange pattern*.

Chapter 5

5 Towards a Security Inter-cloud Reference Architecture

In this Chapter an approach for building a *Security Inter-Cloud Reference Architecture* is presented. A *Security Reference Architecture* can be used to provide a reference for security certification of services. Also, knowing the *misuses* that affect a particular service, a provider can show that their service can handle the corresponding threats by incorporating appropriate *Security Patterns*, which would increase consumer trust in their use.
A threat analysis and three *Misuse Patterns* for *federated Inter-cloud* systems are presented as examples on how to use the *Inter-cloud Reference Architecture* proposed in Chapter 4 as a base for the construction of a *Security Inter-cloud Reference Architecture*.

As stated in Chapter 2 (Background Information), *Misuse Patterns* will be used to improve the design of federated *Inter-Cloud* systems and to perform the security validation of the constructed system. Detecting and blocking the threats in each use case (using *Security Patterns*) will complete the development of a *Security Inter-cloud Reference Architecture* (*Security Patterns* will be used in future works, see Section 7.1).

This Chapter is organized as follows: First, a threat analysis is performed over the activities that relates the Use Cases *Request a Service* (Section 4.1.2.1) and *Create a Service* (Section 4.1.2.2), then over the results of such analysis, is applied the methodology used in [Bra08] to obtain a list of threats. By listing the threats of every Use Case, it is possible to infer or detect some misuse activities that appears in one or more Use Cases to achieve a security violation; these misuse activities could be generalized and be presented in a *Misuse pattern* fashion. Three *Misuses patterns* are identified and presented in the last section of this Chapter, showing how an attacker could perform an attack and the ways to stop, react and mitigate it.

5.1 Identifying threats

The identification of threats is absolutely necessary in order to define and apply proper defenses. It is possible to enumerate threats systematically by considering each activity from each Use Case. Each activity should be analyzed in order to detect possible threats. In this work we will use the methodology proposed in [Bra08] to detect threats.

From the Use Cases listed in Section 4.1.2 it is possible to reveal several threats. In order to simplify the process of exemplification, it is analyzed one Use Case and extract from it one *Misuse Pattern* that will be described later in Section 5.3.

It is possible to enumerate threats systematically by using uses cases and activity diagrams. Previously, it has been identified use cases for cloud environments in [Fer13b]. This systematic approach implies a detailed investigation of each activity from each use case in order to find possible threats [Bra08] which can also be combined with the analysis of security.

Figure 10 describes an activity diagram that involves the Use Cases *Request a Service* and *Create a Service*. Several threats can be revealed from these interactions: for example, when a cloud user (consumer) requests a specific service, but another one is provided (see Section 5.3). The dashed line in the diagram (when the *Service B* is created) indicates where the misuse passes the *service B* as response to the request for service from the *Service Provider 1* and not the *Service A*, that was registered in the *Inter-Cloud* and originally requested.

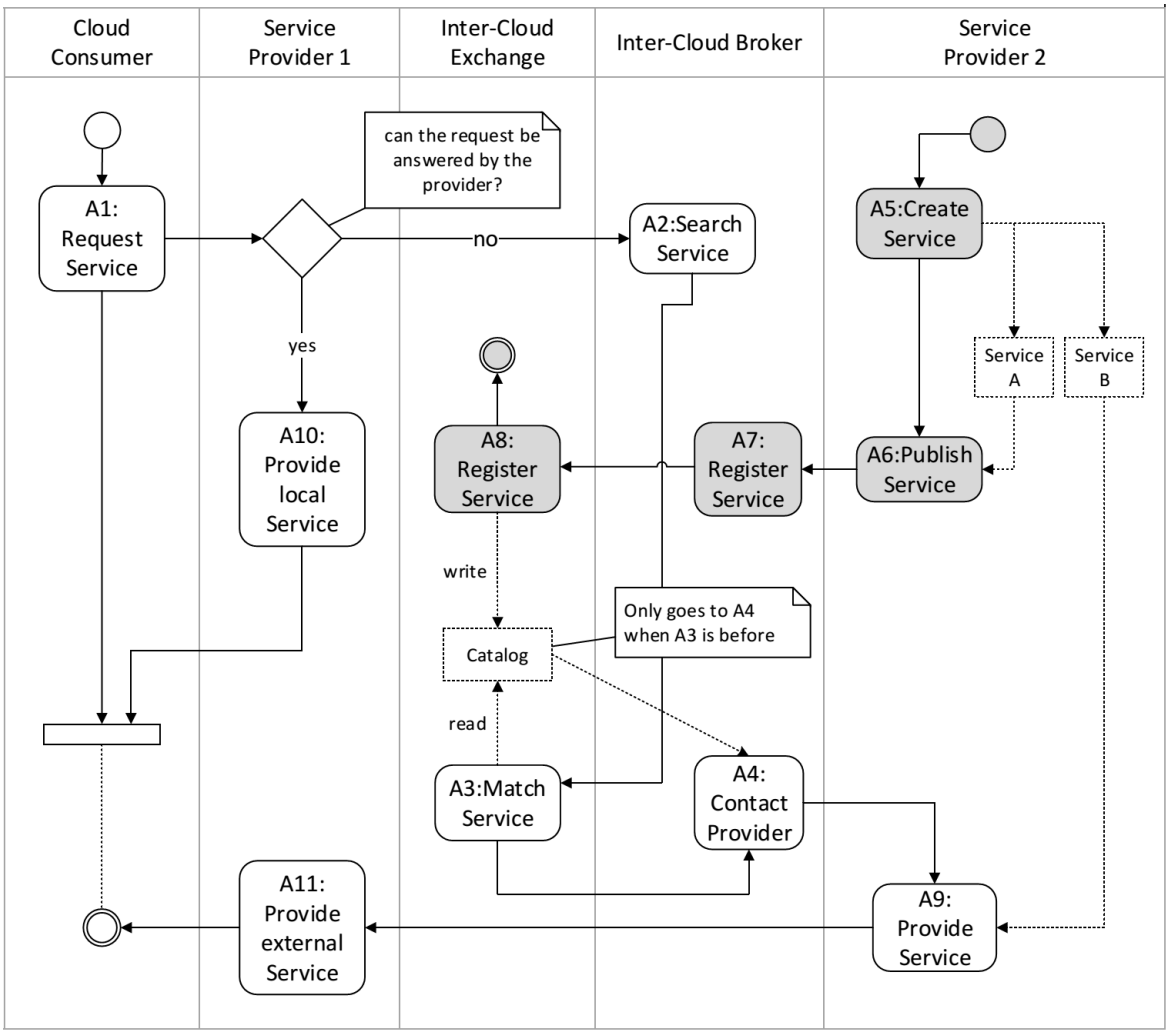


Figure 10: Steal information using a malicious service with Use Cases Request a Service and Create a Service

We have identified some threats by analyzing the flow of events of the Use Cases 1 (*Request a Service)* and 6 (*Create a Service)* (Figure 10). *Table 1* summarizes the analysis of each action in the activity diagram of the figure according to the security attributes which may be compromised, the source of the threat, and the assets that can be compromised. The standard main security attributes are *confidentiality* (*CO*), *integrity* (*IN*), *availability* (*AV*), and *accountability* (*AC*). The source of the threat can be an *authorized insider* (*AIn*), an *unauthorized insider* (*UIn*), or an *outsider* (*Out*).

			Misuse Activity				
Actor	**Action**	**#**	**Sec. Att. CO/IN/AV/AC**	**Source AIn/UIn/Out**	**Description**	**Attacker**	**Asset**
Cloud Consumer	A1:Request Service	T1.1	IN	UIn/Out	Request service(s)	Ext.	Service
		T1.2	CO	Out	Collects sensitive information, traffic analysis	Ext.	Service
		T1.3	CO	Out	Predict behavior, inference	Ext.	Service
		T1.4	CO/IN/AV	AIn/UIn/Out	Contact a malicious Broker	Malicious admin., Ext.	Service
Inter-Cloud Broker	A2: Search Service	T2.1	CO/IN/AV	AIn/UIn/Out	Search in a malicious Market	Malicious admin., Ext.	Service
		T2.2	AV	Out	Disrupt the Search Service	Ext.	Service
		T2.3	CO	AIn/UIn/Out	Information disclosure (what the Providers are looking for)	Malicious admin., Ext.	Service
		T2.4	IN	Out	Spoofing, Impersonate a Service provider	Ext.	Service
Inter-Cloud Exchange	A3:Match Service	T3.1	IN	AIn/UIn/Out	Change matching in order to favoring or harm a Provider	Malicious admin., Ext.	Catalog
Inter-Cloud Broker	A4: Contact Provider	T4.1	IN	AIn/UIn/Out	Contact a malicious Provider	Malicious admin., Ext.	Service
Service Provider2	A5: Create Service	T5.1	IN	AIn/UIn/Out	Create/modify/delete a Service	Malicious admin., Ext.	Service
	A6: Publish Service	T6.1	IN	AIn/UIn/Out	Publish a Service that does not correspond	Malicious admin., Ext.	Catalog
Inter-Cloud Broker	A7: Register Service	T7.1	IN	AIn/UIn/Out	Modify or avoid registration	Malicious admin., Ext.	Catalog
Inter-Cloud Exchange	A8: Register Service	T8.1	IN	AIn/UIn/Out	Modify or avoid registration	Malicious admin., Ext.	Catalog
Service Provider2	A9: Provide Service	T9.1	CO/IN	AIn/UIn/Out	Service is not the requested	Malicious admin., Ext.	Service
Service Provider1	A10: Provide Service	T10.1	CO/IN	AIn/UIn	Accept Services from any Service Provider	Malicious admin., Ext.	Service

Table 1: Misuse Activities Analysis

But, listing threats is not enough; we need a way to understand how an attack happens. *Misuse patterns* describe how an attack happens from the point of view of an attacker. They can be used to check the security of the reference architecture.

5.2 Misuse patterns

In this section are presented three *Misuse patterns* that describe some threats found in *Federated Inter-Cloud* systems.

One of the threats is the possibility that an attacker or an evil SP deceives the *Inter-Cloud* by providing a different service from that proposed. The attacker could replace locally the service previously registered in the *Inter-Cloud Catalog*. The *Inter-Cloud* would show the registered service as available but the new service is the one that is delivered. This malicious service could perform erasing, modifying and creating information; also, their actions could result in leaking sensitive information delivered confidently.

The two remaining misuses affects directly the availability of the *Inter-Cloud*. A *Denial-of-Service misuse pattern* shows how the attacker will try to disrupt the availability of the *Inter-Cloud* system exhausting the resources through many requests for a particular service or by interrupting the search service flooding with messages the *Inter-cloud*. Due to the similarities between them, they are presented as part of the same pattern but being discussed with particularity when correspond.

5.3 Misuse Pattern: Steal information using a malicious service

5.3.1 Intent

An attacker (malicious *Service Provider*) may change or give a different service from the offered initially; therefore, the *Consumer* could give all its data to a malicious service that only wants to steal information.

5.3.2 Context

In an *Inter-Cloud* system, resources are shared by multiple *SPs* that belong to a *Federation*. A *SP* must belong in advance to the *Federation* to publish/request resources. Any resource of the *Federation* is accessible through the Internet. *Cloud Exchange* is the component where every request and assignment is bound. The requested/provided resource can be at the SaaS, PaaS, or IaaS level. *SPs* publish their resources to the *Cloud Exchange* through the *Cloud Broker*. The *Federation* has insecure (or even none) regulation(s) for accepting new *SPs*.

5.3.3 Problem

How can a malicious *SP* deceive the *Consumer*, *Cloud Broker* and *Cloud Exchange* by giving a different service from the offered? It's possible that the malicious *SP* offers a resource that does not complies with the performance offered. Also, it is possible that the *SP* steal information of its *Consumers* by giving a *Service* that maybe comply with the agreement, but also acts maliciously.

The solution is affected by the following forces:

- Objectives– Its objectives may be vandalism, political actions, or monetary gain.
- Silence– As long as the resource can go unnoticed it is better for the attacker; for example, it can obtain a bigger amount of information or be executed by more *Consumers* (since the resource is part of a Federated network and could be shared between several Consumers).

5.3.4 Solution

5.3.4.1 Structure

Service Providers (*SPs*) forms a federation in order to share capabilities (*Inter-Cloud*). It keeps a *Catalog* of the resources available in the *Inter-Cloud* to decide how to satisfy them. The *Inter-Cloud* could be composed by federated *SPs* or by an association through a broker (acting as front-end and single point of access for every *SP*). Most *Inter-Cloud* projects use a centric component where the requests of the *SPs* are satisfied. Compliance and agreements would be specified in a kind of *SLA*. Figure 2 shows a class diagram for the *Federated Inter-Cloud* pattern. A *SP* processes requests from *Consumers* through an *Inter-Cloud Portal*. The *SP* performs requests through an *Inter-Cloud Broker*. The *Inter-Cloud Broker* implemented internally or externally redirects the request to the *Inter-Cloud Exchange* component; both are part of the *Cloud Federation* system. The *Inter-Cloud Exchange* consults its *Catalog* (a.k.a. *Information Repository*) looking for the best match for the request. The *Catalog* lists several kinds of *Services*. The *Inter-Cloud Exchange* is aware of the condition of the entire system through status information coming from *Inter-Cloud Brokers*. The *Catalog* is updated considering the information sent by the *Inter-Cloud Brokers*.

5.3.4.2 Dynamics

UC: The Service is not the requested (actor: Attacker)

- Summary: An Attacker (malicious *Service Provider*) may change or give a different service from the offered initially.
- Precondition: The *Cloud Broker* monitors the information and agreements compliance of the SPs associated to it. A *SP* must belong in advance to the Federation to be able to publish/request resources.
- Description:
    a) The Attacker (malicious *Service Provider*) creates the *Services A* and *B* (*UC.* 6, section 4.1.2.2).

b) The Attacker publishes the *Service A* in the *Federation*. (UC. 12, section 4.1.2.2).
c) A *Consumer* request a *Service* (UC. 1, section 4.1.2.1)
d) The *Attacker* changes the *Service* offered maliciously
e) The *Cloud Exchange* binds the *Service* to the *Consumer* (UC. 19, section 4.1.2.4)
f) The service is provided directly from the *SP* to the *Consumer*.
g) The *Consumer* uses the requested *Service*

- Post condition: The *Consumer* utilizes the malicious *Service* without notice that is not the *Service* requested.

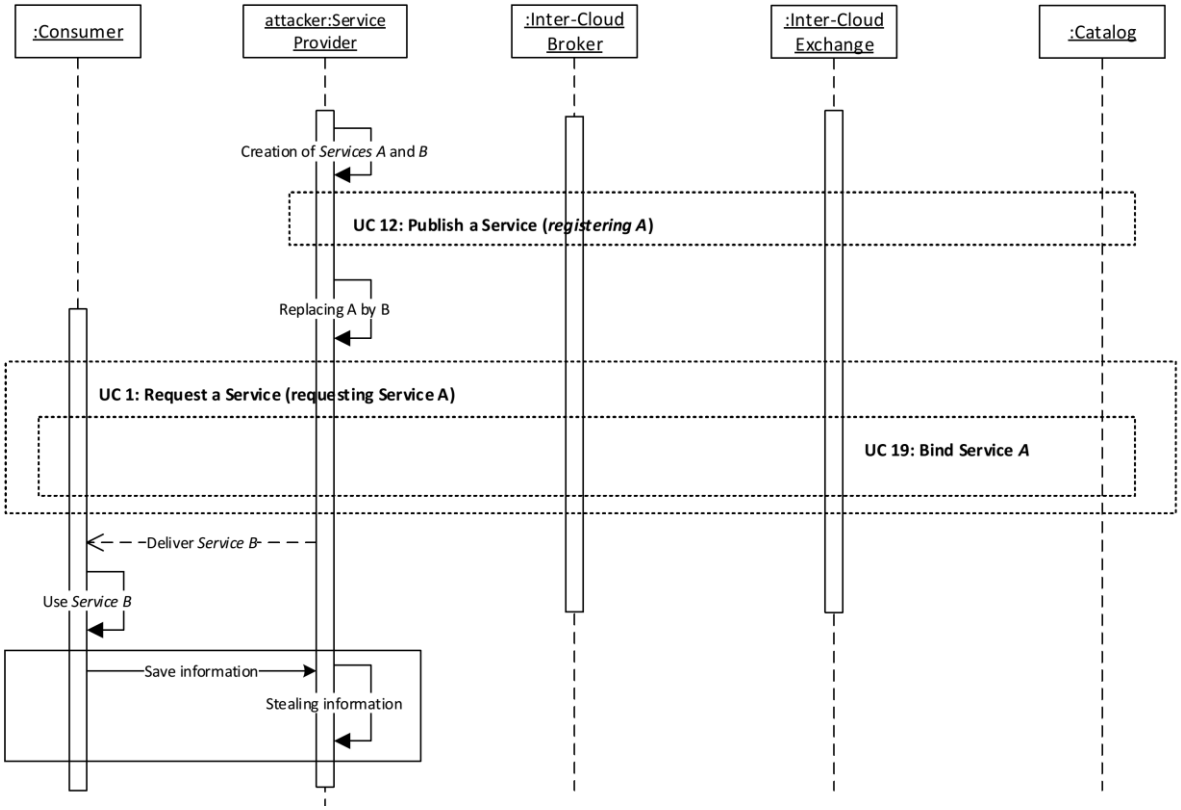


Figure 11: The Service is not the requested

5.3.5 Known uses

Although *Inter-Cloud* system are still under development, several attacks has been employed in *Cloud Computing* systems that are closely related to the *Inter-Cloud* since their purpose, manner and results are the same. For example:

- The service provided could be the instantiation of a *Virtual Machine*, but if, that instantiation is done from a public repository or from an untrusted provider, could lead to a misuse. The malicious modification and publication of a *Virtual Image* (that contains evil code), could incite a data leakage or data modification [Fer13b].
- The *SP* that requests the service must be sure that the service provided is really what was requested or at least, trust the *SP* that provided it. Although it is difficult to realize that a service is not working as promised or that the information was not tampered. The use of multiple and distinct clouds executing the same application can partially solve the problem by comparing the obtained results. The *SP* that requests the service could get evidence on the integrity of the results. This issue and several others are treated in [Boh13].

5.3.6 Consequences

This misuse has the following advantages for the attacker:

- Objectives – Its vandalism objectives can be reached if the malicious SP destroys or modifies the information provided. Its monetary objectives can be reached due the fact that the attacker could sell the information gathered from the consumers. Finally, its political objectives can be reached leaking sensitive information.
- Silent – If the maliciously given *Service* is not detected, it could be shared by every *Consumer* in the *Federation*, making whatever it wants, even steal and modify the *Consumer* information.

Possible sources of failure include:

- The *Service* that performs the request for service is notified by the *Cloud Broker* or *Cloud Exchange* about the *Service* and *SP* that was assigned. This notification should be useful to authenticate and check that the *Service* provided is at least the same registered in the *Federation*.

5.3.7 Countermeasures

To prevent a false service we can use the following countermeasures:

- Authenticate each service by the consumer.
- Reputation systems would help in the selection phase. It is possible to store/process the most critical data in a very trusted *SP*.
- Obfuscate the way the information is stored in the *SP*. In [Boh13] is proposed several alternatives to avoid the disclosure of information. Is possible to diversify the data store in

several *SPs*, even, comparing the outcome information from two or more *SPs* in order to be sure that the process is done correctly.

- Selection- Reputation systems would help in the selection phase. There must exist a selection in terms of trust of the *SPs* that join the Federation. If a single *SP* is untrusted, it puts in risk the entire *Inter-Cloud*.
- Authority to Remove- the *Cloud Exchange* and the *Cloud Broker* must have the authority for removing a *SP* of the *Federation* when they estimate that it is dangerous.
- *Cloud Exchange* should register a kind of signature for the resource that is published along with the performance metrics offered.
- *Cloud Exchange* should send the signature information to the related *Cloud Brokers,* this information must be known also by the *SP* that request the resource.

5.3.8 Related patterns

- Federated Inter-Cloud pattern [Enc14] – solves the problem of resource exhaustion, or the lack of specific services that can affect a SP. The SPs form a federation in order to share capabilities.
- Misuse pattern for Denial-of-Service in federated Inter-Clouds [Enc14b]

5.4 Denial-of-Service in federated Inter-Cloud

5.4.1 Intent

A *DoS* attack may activate the request of many resources, which can exhaust the resources of a Federation, thus denying legitimate users the use of these resources or send a flood of messages to the *Broker* for disrupting the search service between *SPs*.

5.4.2 Context

In an *Inter-Cloud* system, resources are shared by multiple *SPs* that belong to a Federation. A *SP* must join the federation in advance to request resources from them. The requested resource can be at the *SaaS*, *PaaS*, or *IaaS* level. The *Inter-Cloud* is organized around a central component called *Inter-Cloud Exchange*. The *Inter-Cloud Exchange* is where every request and resource assignment are bound. Requests are not made directly to the *Inter-Cloud Exchange,* but must first pass through an *Inter-Cloud Broker*. The Federation has few (or even none) regulation(s) for accepting new *SPs or users*.

5.4.3 Problem

To perform some types of misuse it is necessary to have an account in one or more *SPs* that belong to the Federation. How can the attacker deny access to others consumers? The attacker could request many resources (maybe a high amount of one specific service or many of many kinds), exhausting the resources available to the other consumers. Alternatively, it could generate a flood of messages for disrupting the search service of the *Inter-Cloud Exchange*, making the entire system unable to respond to requests for services.

The solution for the attacker is affected by the following **forces**:

- Objectives – Its objectives may be vandalism, political action, or monetary gain.
- Duration – Usually, the longer the service will be unusable, the better for the objectives of the attacker.
- Untraceability – Since the attack compromises several components and consumers, it should be hard to trace who is the responsible one.
- Obfuscation – Since the attack may compromise several shared resources, all of them would be affected; even if the attacker objective was just one service. If he blocks them all, it could cause consumers and providers confusion (even panic) and will obfuscate its real intentions.

The attack can be performed by taking advantage of the following vulnerabilities:

- Any consumer can open an account in a *SP*.
- Any consumer can request what he needs to its *SP*, even unlimited amounts.
- The *SP* consults automatically the *Inter-Cloud Broker* to see if it does not have enough amount or kind of requested resource. The *Inter-Cloud Broker* will redirect the request to the *Inter-Cloud Exchange*.
- Most providers do not control the number of requests that can be done in a given amount of time.
- *SPs* access *the Inter-Cloud* only through an *Inter-Cloud Broker*.
- The *Inter-Cloud Broker* is responsible of the forwarding of requests coming from the *SPs* to the *Inter-Cloud Exchange* which sends the requests to the *SPs*. Both units are then single points of choke or failure.
- The address of the *Inter-Cloud Broker* is public and reachable for anybody.

5.4.4 Solution

A *Denial-of-Service* (*DoS*) attacker could perform many requests to the *Federation* in order to significantly reduce the availability of the resources (resource exhaustion) with the objective of leaving the rest of the *Consumers* without resources to use. Also, it could flood the *Broker* in order to disrupt the search of services between *SPs* with the objective of making impossible to use the *Inter-Cloud*.

5.4.4.1 Structure

Figure 2 shows a class diagram of a **Cloud Federation**. A **Service Provider** processes requests from **Consumers** through a **Portal**. The **Service Provider** forwards user requests to an **Inter-Cloud Broker** when resources are needed. The **Inter-Cloud Broker** redirects the requests to the **Cloud Inter-Cloud Exchange**. The **Inter-Cloud Exchange** consults its **Catalog** (a.k.a. *Information Repository*) looking for the best match for the request. The **Catalog** lists the resource amounts and types of services. The **Inter-Cloud Exchange** is aware of the conditions of the entire system through status information coming from **Inter-Cloud Brokers**. The **Catalog** is constantly updated using the information sent through the **Inter-Cloud Brokers**.

5.4.4.2 Dynamics

We show here two misuses representing the previous solution; both use cases make use of the F*ederated Inter-Cloud* pattern [Enc14]:

- *UC: E*xhaust resources through many requests (actor: Attacker) (Figure 12)
  Summary: An **Attacker** (probably through many zombies) performs many requests to its **Service Provider** in order to slow down the **Inter-Cloud Exchange**, thus impeding and delaying the requests from other **Consumers** in the Federation.
  Actor**: Attacker**
  Precondition: The **Attacker** (and his zombies) must have an account in one or more **Service Providers** that belong to the Federation.
  Description:
  a) The **Attacker**, directly or through his zombies, performs multiple requests to its **Service Provider**.
  b) The **Service Provider** redirects the requests to the **Inter-Cloud Broker**, which in turn sends them to the **Inter-Cloud Exchange**.
  c) The requests take up most resources of the **Inter-Cloud Exchange** and requests from other users are denied.

Post condition: **Inter-Cloud Exchange** or **Catalog** take too long to respond other consumers' requests, or even might not be able to answer them due to resource exhaustion.

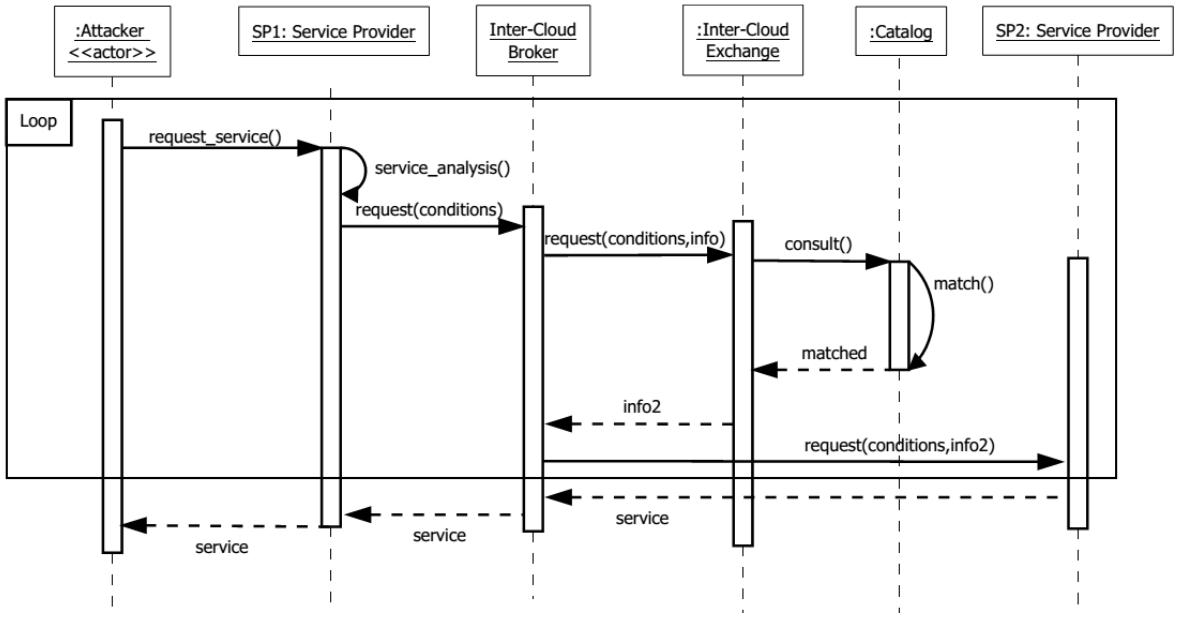


Figure 12: Exhaust resources through many request

- *UC:* Disrupt the search of services (actor: Attacker) (Figure 13)

  Summary: An **Attacker** floods with messages the **Inter-Cloud Broker**, disrupting the search of services from the **Inter-Cloud Exchange**.

  Actor**: Attacker**

  Precondition: The **Inter-Cloud Broker** forwards the request of services of the **Service Providers** associated to it. The address of the **Inter-Cloud Broker** is public and reachable for anybody.

  Description:

  a) The **Attacker** floods the **Inter-Cloud Broker** with messages.

  b) The **Inter-Cloud Broker** tries to forward the request for service between **Service Provider** and **Inter-Cloud Exchange**.

  Post condition: The **Inter-Cloud Broker** failed to forward the request for services and the users cannot access the requested services.

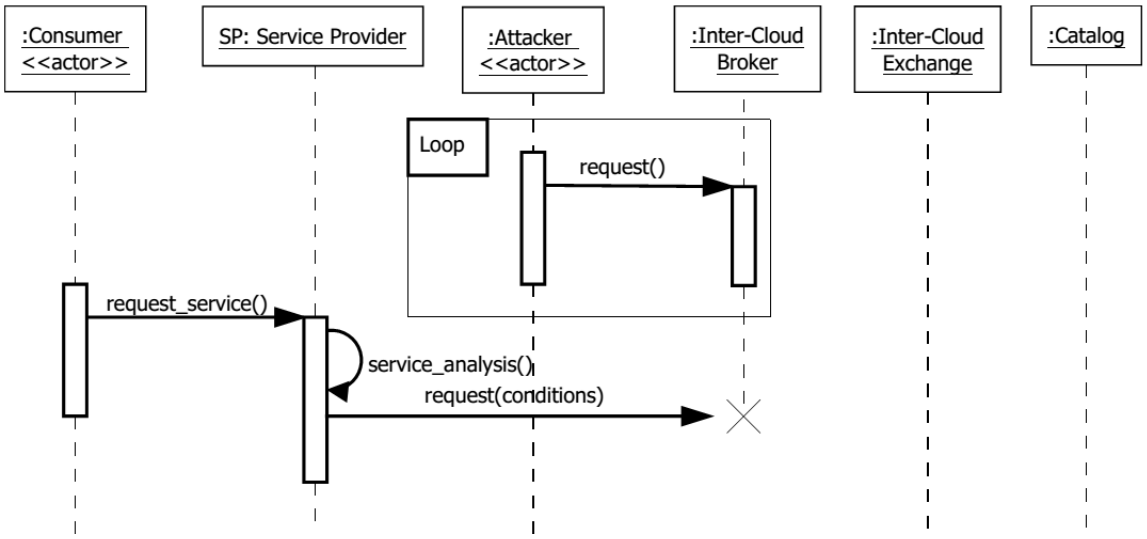


Figure 13: Disrupt the search of services

5.4.5 Known uses

Attacks in *federated Inter-Cloud* environments have not happened yet since the *Inter-Cloud* is very new and is still under development. However, we should keep in mind the possible vulnerabilities of the system in order to produce more secure designs.

5.4.6 Consequences

This misuse has the following advantages for the attacker:

- Objectives – Its vandalism objectives can be reached if the *SP* allows multiple requests. If the *SP* provides an *API*, it is possible for the attacker to automate the attack with a script that can request resources for an arbitrary time. Its political objectives can be reached, for example, if he plans his attack the days before elections (the attacker can launch the attack when he considers it convenient). Its monetary objectives can be reached because the attacker can launch the attack whenever he wants (same as the previous objective); for example, threaten to shut down the federation the day before Christmas unless he is paid with some amount of money.
- Duration – The attacker can plan the duration of the attack and even keep attacking when the system is reestablished.
- Untraceable – Since anyone can get an account in a *SP* or launch an attack from an anonymous place, the attack might be untraceable.
- Obfuscation – Since every shared resource is affected, the real intention of the attacker can be obfuscated making it more difficult to stop or take some particular measure against the attack.

Possible sources of failure include:

- A limit on the amount of resources that can be requested by the consumer or **SP** (through policies).

  Passing through a broker and not directly to the **Cloud Exchange** can potentially be an advantage for the defender.

5.4.7 Countermeasures

Denial-of-Service in *federated Inter-Cloud* can be stopped by the following countermeasures:

- Have a policy about the amount of resources that can be requested.
- Filter requests- The **Inter-Cloud Broker** must analyze and filter all requests to determine if they are appropriate in type and quantity.
- Replication- Replication must be part of the design of the **Inter-Cloud Exchange**, **Inter-Cloud Broker** and **Catalog** component. Under a *DoS* attack, the components must stay always online; this also will avoid scaling problems and will improve fault tolerance.
- *IDS-Firewall- IDSs* and firewalls can ensure that packets with very large sequence numbers and garbage packets are discarded. Again, the *IDS* pattern [Fer13a] is relevant, as well as the Firewall patterns [Sch06].
- Use of *Proxy* and *Stateful Firewalls* [Sch06], which can look inside the request packets and analyze their contents, as well as the headers, to decide if the information is appropriate or not. These can be implemented within the *Broker*.

5.4.8 Forensics

Where can we find evidence of this attack?

- The *Broker* is the first component that should be analyzed, *IDS* and *Firewall* patterns can help in the forensic task.
- *Cloud Exchange* must receive request information from the *Broker*, every request made must be logged, and also every time a *SP* joins the Federation the *Broker* must inform it to the *Cloud Exchange*.
- *Cloud Exchange* must log all the information about the assignment corresponding to a request.

5.4.9 Related Patterns

- *VoIP misuse patterns*- A *Voice over IP DoS attack* is presented in [Pel09] by overwhelming resources in order to disrupt *VoIP* operations, typically through a flood of

messages; that is an attack that interrupts the way the agreements compliance are measured or controlled (not just disrupt the normal operation of the services).

- *Abstract IDS pattern* – it allows monitoring of all traffic as it passes through a network, analysis the traffic to detect possible attacks, and trigger an appropriate response [Fer13a].
- Firewall patterns like *Packet Filter Firewall* and *Proxy-Based Firewall* [Sch06].
- *Federated Inter-Cloud pattern* [Enc14] – solves the problem of resource exhaustion, or the lack of specific services that can affect at one SP. The SPs form a federation in order to share capabilities. Under a *DoS* attack the resource exhaustion affect the entire Federation.

Chapter 6

6 Discussion

It was possible to abstract a *Reference Architecture* which covers the most important proposals in the federated *Inter-Cloud* environment. To the best of our knowledge, the *Inter-Cloud Reference Architecture* presented in this thesis is the first proposal of a *Reference Architecture* for the *Inter-Cloud* environment. These results should help in the design phase of new *Inter-Cloud* proposals, establishing a common ground in terminology, relations, and functions. Although this is a concrete contribution to the State of the Art in the *Inter-Cloud* environment, this also, proposes the possibility of using the same model as a basis into the establishment of a *Security Inter-Cloud Reference Architecture*.

This *RA* is much more comprehensive and detailed than other specific projects proposals, and it is not attached to concrete technologies, protocols or tools. Although the *RA* is a generalization of several current proposals, it provides a higher level of detail, since is possible to establish the multiplicity of relationships and also have a clear perspective of the interactions because the solutions are expressed with Use Cases templates, static and dynamic diagrams, a better approach than the commonly used block diagrams.

The use of UML as a tool for expressing the solutions also gave us the opportunity to express them in a well-defined language that can be easily understood by the majority of people related to IT and computer science.

The *RA* was constructed using architectural patterns and also performing an analysis that contemplated listing actors, their functions, and their relationships. The use of patterns into the process for defining the *RA* allowed us to capture and express the experience of experts about good practices and document the same in an accessible and understandable way for designers and non-security experts. Patterns describe good designs in a way that makes it possible for others to understand and reuse them. The final *RA* is a composition of several patterns, each accomplishing and isolating specific problems and solutions, and because of that, the solution is also very maintainable and easy to understand.

The current *RA* was used as base for the creation of several *Misuse Patterns*. Every of these patterns represents an attack that could be stopped by the correct application of *Security Patterns* or some policies.

These *Misuse Patterns* [Fer13a] can help designers to understand how an attack is performed and what components of the system were used and compromised during an attack. They follow a template that contain different sections: ***context*** that describes the environment where the attack is performed, ***problem*** that includes forces which describes the vulnerabilities of the system, and ***solution*** that depicts UML diagrams illustrating how the attack is performed.

Misuse Patterns also describe the consequences which discuss the benefits and drawbacks of the pattern from the attacker's perspective, countermeasures that list security mechanisms which can be applied to stop or mitigate the threat, and forensics that indicate how to trace an attack once it happens or where to search for forensic data. It is possible to build a relatively complete catalog of *Misuse Patterns* for *Inter-Cloud*s. Having such a catalog it would be possible to analyze a specific *Inter-Cloud* architecture and representing the system using a class diagram, analyze how an attack is performed using sequence diagrams, and evaluate its degree of resistance to these misuses. The architecture (existing or under construction) must have a way to prevent or at least mitigate all the misuses that apply to it.

Some projects proposed a complete decentralized architecture for the *Inter-Cloud* (without a central component), a style more like in a peer-to-peer fashion. But, these projects are still under development and there are not enough proposals in the same direction, limiting the material to extract a pattern from them and add them to the current proposal.

An important aspect in security which is also considered in several *Inter-Cloud* projects is the existence and utilization of a Trust index for measuring and performing a better match for a *SP*. Trust issues will be considered in future work where it will be necessary to apply some *Security Patterns* to the current solution. In the same way that Trust, compliance with Service Level Agreements is also a subject that will be approached when a complete *Security Inter-Cloud Reference Architecture* has been produced.

Chapter 7

7 Conclusions

Inter-cloud systems are complex systems that leverage different technologies and can be deployed in different ways, as well as provide different types of services. All this implies that it can be a challenge to understand how to design a suitable system.

7.1 Contributions

During the course of this thesis the following contributions have been produced:

- Three *Inter-Cloud* architectural patterns to design and understand the properties of this kind of systems, their most important components and relationships.
- Three *Misuse patterns* as the first step into the building of a *Security Inter-Cloud Reference Architecture*, which also serve for evaluating and validating the security of an *Inter-Cloud* system.
- Synthesis and abstraction of works related to *Inter-Cloud*, generating a language for federated *Inter-Cloud* systems.
- An *Inter-Cloud Reference Architecture* has been developed using the *Inter-cloud* patterns mentioned earlier as well as an analysis of the system, characterizing its stakeholders, roles, and most important use cases. To the best of our knowledge this is the first *Inter-Cloud Reference Architecture* published.
- The proposed *Inter-Cloud Reference Architecture* is more understandable than other specific works about *Inter-Clouds* because it is not based on specific protocols or tools.
- Due to this *Inter-Cloud Reference Architecture* was built upon patterns, and patterns is almost always an ongoing work, the development of the *RA* for the *Inter-Cloud* could be improved just having into account new patterns that comply better the requirements of the system or easily extended by aggregating new components.

7.2 Publications

During the development of this thesis, three scientific papers were published in international conferences:

- O. Encina, E. B. Fernandez and R. Monge. "A misuse pattern for Denial-of-Service in federated Inter-Clouds". *Proceedings of 3rd Asian Conference on Pattern Languages of Programs*, Tokyo, Japan, March 2014.
- O. Encina, E. B. Fernandez and R. Monge, "Towards Secure Inter-Cloud Architectures". *Proceedings of Nordic pattern conference on Pattern Languages of Programs,* Sagadi Manor, Estonia, April 2014.
- O. Encina, E. B. Fernandez and R. Monge. "Threat analysis and misuse patterns of federated Inter-Cloud systems". *Proceedings of 19th European Conference on Pattern Languages of Programs*, Bavaria, Germany, July 2014.

7.3 Summary

The idea of an *Inter-cloud* opens up a wide range of interesting research topics. The building of a *Reference Architecture* for this kind of system using patterns lead to the following conclusions:

- The *Reference Architecture* showed helps in the possible implementation of future *Inter-Cloud* systems by indicating its critical and most fundamental aspects that could be complemented with *Security Patterns* in order to achieve *a Security Inter-Cloud Architecture*.
- *Misuse Patterns* are also useful to evaluate existing *Inter-cloud* systems by examining them to see if they contain the required *Security Patterns*, providing a way to evaluate the complete system by checking their components and relationships. We can apply them for example to evaluate the security of a specific *Inter-cloud* system once the *Security Inter-cloud Reference* has been built.
- The use of patterns in this kind of projects helps in the unification of ideas, and serves as a guide for future developments, improving the way these are designed, taking into account a set of considerations while avoiding common mistakes, emphasizing the holistic thinking.
- The *Reference Architecture* and the patterns presented like a whole provides a set of recommendations to consider that would facilitate the success of the system, acting like catalyzers to allow a flow of comprehensive information between every actor.
- The use of patterns for building a *Reference Architecture* allow the construction of systems easily, if patterns are mixed with others a clear guide of the advantages and disadvantages could be obtained, improving the way the design of the system is addressed.
- In the near future, it can be expected that hundreds of cloud providers will compete to offer services and thousands of users also compete to receive the services to run their complex heterogeneous applications on *Cloud Computing* environment. The use of a

Reference Architecture will improve the development and integration between multiple cloud systems.

7.4 Future Work

Future work will focus on the creation of a *Security Inter-Cloud Reference Architecture*. The basic approach that will be used to build it is by applying a systematic methodology from [Fer14a], which can be used as a guideline to build secure cloud systems and to evaluate their security levels. *Misuse Patterns* for the *Inter-Cloud* will be proposed in order to validate the proposal (e.g. Inter-Cloud Service Hijacking or Malicious Service Offering misuse patterns). In this direction, it will be necessary to analyze threats and apply the corresponding countermeasures realized as *Security Patterns* (e.g. Inter-Cloud Security Logger and Auditor, or Inter-Cloud Remote Authenticator/Authorizer patterns) over the *RA* previously presented by examining the actions of every Use Case looking for threats.

It will be possible to evaluate the degree of security of every *Inter-Cloud* abstract and concrete architecture and performs specific tests using the *Misuses* previously presented and those that will be developed to complete the catalog.

In regard to the *Inter-Cloud* technology, there are efforts to solve problems related with the Trust, might be is possible to establish a Trust model which consider a dynamic indicator who depends of the history of the *SP* and its current behavior.

Another problem to solve is how to manage the *Catalog* data, current proposals propose the distribution of data between several components using a peer-to-peer Distributed Hash Tables (DHT). Finally, a recurrent problem is how to find the best *SP* to respond to a resource request. Due to the extensive list of resources and *SPs*, it might be necessary to use heuristic models who can resolve as fast as it can the *SP* who best suits to the requirements of the requestor.

With respect to the dissemination of this work, it is contemplated the publication in an international Journal the results of this thesis, and it would also be possible the publication of the complete *Security Inter-Cloud Reference Architecture* proposal.

8 References

[Avg03] P. Avgeriou. "Describing, instantiating and evaluating a reference architecture: A case study". In *Enterprise Architect Journal. Fawcette Technical Publications*, June 2003.

[Ber09] D. Bernstein, E. Ludvigson, K. Sankar, S. Diamond, M. Morrow. "Blueprint for the Intercloud - Protocols and Formats for Cloud Computing Interoperability". *In Internet and Web Applications and Services*, 2009. *ICIW '09. Fourth International Conference* on, 24-28 May 2009, Venice, Italy.

[Ber10] D. Bernstein and D. Vij, "Intercloud security considerations", In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, IEEE, 2010, 537–544.

[Ber11] D. Bernstein, D. Vij, and S. Diamond. "An intercloud cloud computing economy-technology, governance, and market blueprints". In *SRII Global Conference (SRII)*, 2011 Annual, pages 293–299. IEEE, 2011.

[Boh13] J. Bohli, N. Gruschka, M. Jensen, L. Iacono, and N. Marnau. "Security and Privacy-Enhancing Multicloud Architectures". *IEEE Trans. Dependable Secur. Comput.* 10, 4 (July 2013), 212-224.

[Bra08] F. Braz, E. B. Fernandez and M. VanHilst, "Eliciting Security Requirements through Misuse Activities", *Proceedings of the 2nd International Workshop on Secure Systems Methodologies using Patterns (SPattern'07)*.In conjunction with the *4th International Conference onTrust, Privacy & Security in Digital Business (TrustBus'07)*, Turin, Italy, September 1–5, 2008, 328–333.

[Bus96] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal., *Pattern- oriented software architecture*, Wiley 1996.

[Buy09a] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility". In *Future Generation computer systems*, 2009, 25(6):599–616.

[Buy09b] R. Buyya, R. Ranjan and R.N. Calheiros. "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and

opportunities", *High Performance Computing & Simulation, 2009. HPCS '09. International Conference on*, vol., no., pp.1,11, 21-24 June 2009

[Buy10] R. Buyya, R. Ranjan, and R. N. Calheiros, "Intercloud: Utility-oriented federation of Cloud computing environments for scaling of application services". In *algorithms and architectures for parallel processing*, Springer, 2010, 13–31.

[Cal12] N. M. Calcavecchia, A. Celesti and E. D. Nitto. "Understanding Decentralized and Dynamic Brokerage in Federated Cloud Environments." *Achieving Federated and Self-Manageable Cloud Infrastructures: Theory and Practice. IGI Global*, 2012. 36-56.

[Car12] E. Carlini, M. Coppola, P. Dazzi, L. Ricci and G. Righetti, "Cloud federations in contrail". *In Euro-Par 2011: Parallel Processing Workshops*, pages 159–168. *Springer*, 2012.

[Cel10a] A. Celesti, F. Tusa, M. Villari and A. Puliafito. "How to Enhance Cloud Architectures to Enable Cross-Federation," *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference* on, vol., no., pp.337, 345, 5-10 July 2010.

[Cel10b] A. Celesti, F. Tusa, M. Villari and A. Puliafito. "Security and Cloud Computing: InterCloud Identity Management Infrastructure," *Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE), 2010 19th IEEE International Workshop on*, vol., no., pp.263, 265, 28-30 June 2010.

[Del07] N. Delessy, E. F. Fernandez and M. Larrondo-Petrie. "A Pattern Language for Identity Management". In *Computing in the Global Information Technology, International Multi-Conference on*, (0) 31, IEEE Computer Society, Los Alamitos, CA, USA, 2007.

[Dem12] Y. Demchenko, M. X. Makkes, R. Strijkers, C. Laat; "Intercloud Architecture for Interoperability and Integration", *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference* on, vol., no., pp.666, 674, 3-6 Dec. 2012.

[Enc14a] O. Encina, E. B. Fernandez and R. Monge, "A misuse pattern for Denial-of-Service in federated Inter-Clouds". *Procs. of AsianPLoP (Pattern Languages of Programs) 2014*, Tokyo, Japan, March 2014.

[Enc14b] O. Encina, E. B. Fernandez and R. Monge, "Towards Secure Inter-Cloud Architectures". *Procs. of VikingPLoP (Pattern Languages of Programs) 2014*, Sagadi Manor, Estonia, April 2014.

[Enc14c] O. Encina, E. B. Fernandez and R. Monge, "Threat analysis and misuse patterns of federated Inter-Cloud systems". *Procs. of EuroPLoP (Pattern Languages of Programs) 2014*, Bavaria, Germany, July 2014.

[Enc14d] O. Encina, E. B. Fernandez and R. Monge, "An Inter-cloud Broker pattern". *Submitted for publication*, 2014.

[Enc14e] O. Encina, E. B. Fernandez and R. Monge, "An Inter-cloud Exchange pattern". *Submitted for publication*, 2014.

[Fer13a] E. B. Fernandez, "Security patterns in practice: Building secure architectures using software patterns". *Wiley Series on Software Design Patterns*. 2013.

[Fer13b] E. B. Fernandez, K. Hashizume and M. M. Larrondo-Petrie. "A reference architecture for cloud computing". Submitted for publication, 2014.

[Fer14a] E. B. Fernandez, R. Monge and K. Hashizume, "A Security Reference Architecture for cloud systems", submitted for publication, 2014.

[Fer14b] E. B. Fernandez and S. Mujica, "Two patterns for HIPAA regulations". *Procs. of AsianPLoP (Pattern Languages of Programs) 2014*, Tokyo, Japan, March 2014.

[Gro12] N. Grozev and R. Buyya, "Inter-cloud architectures and application brokering: taxonomy and survey". *Software: Practice and Experience*, 2012, 44: 369-390.

[Jra12] F. Jrad, J. Tao and A. Streit. "Sla Based Service Brokering In Intercloud Environments", *CLOSER 2012- 2nd International Conference on Cloud Computing and Services Science*, 2012.

[Kec12a] G. Kecskemeti, M. Maurer, I. Brandic, A. Kertesz, Z. Nemeth, and S. Dustdar, "Facilitating self-adaptable inter-cloud management", In *Parallel, Distributed and Network-Based Processing (PDP), 2012, 20th Euromicro International Conference*, IEEE, 2012, 575–582.

[Kec12b] G. Kecskemeti, A. Kertesz, A. Marosi and P. Kacsuk. “Interoperable Resource Management for Establishing Federated Clouds.” *Achieving Federated and Self-Manageable Cloud Infrastructures: Theory and Practice.* IGI Global, 2012. 18-35.

[Kle05] L. Kleinrock, “A Vision for the Internet”. *ST Journal of Research 2*(1), 4–5 (2005)

[Ker12] A. Kertesz, G. Kecskemeti, Z. Nemeth, M. Oriol and X. Franch, "A holistic service provisioning solution for Federated Cloud infrastructures," *Software Services and Systems Research - Results and Challenges (S-Cube), 2012 Workshop on European* , vol., no., pp.25,26, 5-5 June 2012.

[Kre11] M. Kretzschmar, M. Golling. "Security management spectrum in future multi-provider Inter-Cloud environments — Method to highlight necessary further development*," Systems and Virtualization Management (SVM)*, 2011 5th *International DMTF Academic Alliance Workshop on, vol., no., pp.1,8, 24-24 Oct. 2011.*

[Meh10] Mohammad Mehedi Hassan, Biao Song, and Eui-Nam Huh. “Horizontal dynamic cloud collaboration platform: Research opportunities and challenges”. *In Proc. of the 2nd Annual International Conference on Cloud Computing and Virtualization (CCV)*, Malaysia, 2010.

[NIS13] NIST Cloud Computing Security Working Group. “NIST cloud computing security reference architecture”, *NIST SP 500-299 (draft)*, 2013 (http://www.nist.gov/itl/csd/cloud-061113.cfm)

[Ngo12] Canh Ngo, Y. Demchenko, and C. de Laat. “Toward a Dynamic Trust Establishment approach for Multi-provider Intercloud Environment”. *In IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom 2012)*, pages 532–538, 2012.

[Red07] S. T. Redwine Jr. Software Assurance: “A Curriculum Guide to the Common Body of Knowledge to Produce, Acquire and Sustain Secure Software”. *Technical Report Version 1.2, US Departments of Homeland Security*, October 2007.

[SIIF] ICWG/2302_WG - Intercloud WG (ICWG) Working Group, “Draft Standard for Intercloud Interoperability and Federation (SIIF)”, *IEEE P2302/D0.2*, January 2012 (http://standards.ieee.org/develop/project/2302.html)

[Sin13] Mukesh Singhal, Santosh Chandrasekhar, Tingjian Ge, Ravi Sandhu, Ram Krishnan, Gail-Joon Ahn, and Elisa Bertino. "Collaboration in multicloud computing environments: Framework and security issues". *Computer*, 46(2):76–84, 2013.

[Sot11] S. Sotiriadis, N. Bessis, N. Antonopoulos, "Towards Inter-cloud Schedulers: A Survey of Meta-scheduling Approaches," In *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, 2011 International Conference on , vol., no., pp.59,66, 26-28 Oct. 2011.

[Wan12] J. K. Wang, J. Ding and T. Niu. "Interoperability and Standardization of Intercloud Cloud Computing". *arXiv preprint arXiv:1212.5956*, 2012.